Inside Mac OS X

La terminologie d'AppleScript Studio

pour AppleScript Studio 1.3

trad.applescript.free.fr
23 février 2004

Préambule

Ce guide n'est absolument pas une traduction officielle de la Société Apple.

Ce guide est basé sur le guide "*AppleScript Studio Terminology Reference for AppleScript Studio 1.3*".

Ce guide a été entièrement composé et produit avec un logiciel libre et gratuit, une distribution de LaTeX 2_ε. Pour ceux qui ne connaissent pas, essayez-le, malgré son aspect austère et déroutant au premier abord, les documents obtenus dépassent largement les traitements de texte conventionnels, la preuve ici. :-))

Ce guide n'est pas exempt d'erreurs de frappe ou d'interprétation et je vous prie de m'en excuser. Si vous constatez des erreurs ou des oublis, je vous remercie de bien vouloir me faire remonter l'information à cette adresse : `trad.applescript@free.fr` afin de les corriger dans une prochaine version.

Dans l'espoir que cette version française comblera l'attente de tous les utilisateurs francophones, je vous souhaite une bonne découverte de la terminologie d'AppleScript Studio.

Nicolas

Marques déposées.

Apple, le logo Apple, AppleScript, Finder, Mac, Macintosh et PowerBook sont des marques déposées de Apple Computer Inc.

Toutes les autres marques sont la propriété de leurs détenteurs respectifs.

Table des matières

Préambule **I**

I Introduction **1**

1 À propos de ce guide **3**

Ce que ce guide contient 4

Ce que ce guide ne contient pas 5

Autres documentations 5

2 Les bases de la terminologie **7**

Information sur les versions 8

Construire des applications AppleScript Studio 9

Les sources de la terminologie d'AppleScript Studio 11

Les suites 12

Comment sont organisées les suites ? 12

La terminologie fournie par le framework Cocoa Application 13

La terminologie fournie par AppleScript Studio 14

Les formes-clés standards 15

Les conventions d'appellation des méthodes et des gestionnaires . . 16

Accéder aux propriétés et aux éléments 17

Les paramètres des gestionnaires d'Events 19

Connecter les gestionnaires d'Events gérant le clavier et la souris . 20

Les messages d'erreur de scripting 20

Utiliser les exemples de scripts 21

"Panels" contre "Dialogs" et "Windows" 21

Un mot sur Unicode Text 22

II Application Suite 25

1 Classes 29
application 29
bundle 38
data 45
default entry 45
event 50
font 54
formatter 56
image 58
item 60
movie 63
pasteboard 64
responder 68
sound 70
user-defaults 72
window 76

2 Commandes 93
call method 94
center 98
hide 99
load image 100
load movie 104
load nib 106
load sound 107
localized string 109
log 111
path for 112
register 116
select 117

select all 117
show 118
size to fit 119
update 119

3 Events **121**
activated 123
awake from nib 123
became key 127
became main 128
closed 129
deminiaturized 130
document nib name 130
exposed 131
idle 132
keyboard down 133
keyboard up 134
launched 135
miniaturized 136
mouse down 137
mouse dragged 138
mouse entered 139
mouse exited 139
mouse moved 140
mouse up 141
moved 142
opened 143
open untitled 143
resigned active 145
resigned key 145
resigned main 146
resized 147
right mouse down 147
right mouse dragged 149

right mouse up . . . 150
scroll wheel . . . 151
should close . . . 151
should open . . . 152
should open untitled . . . 153
should quit . . . 154
should quit after last window closed . . . 155
should zoom . . . 156
shown . . . 157
updated . . . 158
was hidden . . . 159
was miniaturized . . . 159
will become active . . . 160
will close . . . 161
will finish launching . . . 161
will hide . . . 163
will miniaturize . . . 164
will move . . . 164
will open . . . 165
will quit . . . 166
will resign active . . . 167
will resize . . . 167
will show . . . 169
will zoom . . . 169
zoomed . . . 171

4 Énumérations **173**
alert return values . . . 174
alert type . . . 174
bezel style . . . 175
border type . . . 175
box type . . . 176
button type . . . 177
cell image position . . . 177

cell state value . . . 178
cell type . . . 179
color panel mode . . . 179
control size . . . 180
control tint . . . 180
drawer state . . . 181
event type . . . 181
go to . . . 183
image alignment . . . 183
image frame style . . . 184
image scaling . . . 184
matrix mode . . . 185
quicktime movie loop mode . . . 185
rectangle edge . . . 186
scroll to location . . . 186
sort case sensitivity . . . 187
sort order . . . 187
sort type . . . 188
tab state . . . 188
tab view type . . . 188
text alignment . . . 189
tick mark position . . . 190
title position . . . 190

III Container View Suite 193

1 Classes 197

box . . . 197
clip view . . . 202
drawer . . . 204
scroll view . . . 213
split view . . . 218
tab view . . . 221
tab view item . . . 227

view 229

2 Commandes **239**

close drawer 239

lock focus 240

open drawer 240

unlock focus 241

3 Events **243**

bounds changed 243

resized sub views 244

selected tab view item 245

should select tab view item 246

will resize sub views 247

will select tab view item 248

IV Control View Suite **249**

1 Classes **253**

action cell 254

button 254

button cell 261

cell 264

color well 271

combo box 274

combo box item 279

control 280

image cell 284

image view 285

matrix 289

movie view 296

popup button 301

progress indicator 306

secure text field 311

secure text field cell 314
slider 315
stepper 320
text field 325
text field cell 330

2 Commandes **333**
animate 333
go 334
highlight 335
increment 336
pause 337
perform action 337
play 338
resume 339
scroll 339
start 340
step back 341
step forward 342
stop 343
synchronize 343

3 Events **345**
action 345
begin editing 346
changed 348
clicked 348
double clicked 349
end editing 350
selection changed 351
selection changing 352
should begin editing 353
should end editing 354
will dismiss 355

will pop up 356

V Data View Suite 359

1 Classes 363

browser 363
browser cell 370
data cell 372
data column 376
data item 379
data row 384
data source 386
outline view 392
table column 398
table header cell 401
table header view 402
table view 403

2 Commandes 417

append 417
item for 420

3 Events 423

cell value 424
change cell value 426
change item value 427
child of item 428
column clicked 430
column moved 430
column resized 431
item expandable 432
item value 434
number of browser rows 436
number of items 437

number of rows 439
should collapse item 440
should expand item 441
should select column 442
should select item 443
should select row 444
should selection change 445
will display browser cell 446
will display cell 447
will display item cell 449
will display outline cell 450

VI Document Suite 453

1 Classes 457
document 457

2 Events 465
data representation 465
load data representation 467
read from file 469
write to file 470

VII Drag and Drop Suite 473

1 Classes 477
drag info 477

2 Events 481
conclude drop 481
drag 483
drag entered 483
drag exited 484
drag updated 485

drop 486
prepare drop 488

VIII Menu Suite 491

1 Classes 495
menu 495
menu item 499

2 Events 505
choose menu item 505
update menu item 506

IX Panel Suite 509

1 Classes 513
alert reply 513
color-panel 514
dialog reply 518
font-panel 520
open-panel 522
panel 526
save-panel 529

2 Commandes 535
close panel 535
display 537
display alert 540
display dialog 543
display panel 547
load panel 548

3 Events 551
alert ended 551
dialog ended 552

panel ended 553

X Plugin Suite 555

1 Classes 559
plugin 559

2 Events 561
plugin loaded 561

XI Text View Suite 565

1 Classes 569
text 569
text view 569

XII Annexes 579

A Les applications distribuées avec AppleScript Studio 581
Archive Maker 582
Unit Converter 583
Table Sort 584
Save Panel 586

Index 589

Liste des illustrations

2.1 L'instance File's Owner (représentant l'objet application) dans Interface Builder 30
2.2 Le panel "Font" dans Interface Builder 55
2.3 Un "Number Formatter" dans Interface Builder 56
2.4 La fenêtre Info d'un Number Formatter dans Interface Builder . 57
2.5 L'image de l'icone de l'application dans l'onglet "Images" de la fenêtre MainMenu.nib dans Interface Builder 59
2.6 Les fichiers son dans l'onglet "Sounds" d'une fenêtre Nib dans Interface Builder 70
2.7 Une fenêtre 76
2.8 La liste des fichiers du panneau "Groups & Files" dans un projet "AppleScript Application" 77
2.9 La fenêtre Info d'Interface Builder, montrant les informations AppleScript de l'instance "File's Owner" d'une application 125
3.1 Des "boxes", utilisés comme séparateur horizontal et vertical . . 198
3.2 Une fenêtre avec un tiroir ouvert (extrait de l'application "Drawer") 204
3.3 Le panneau "Cocoa-Windows" d'Interface Builder, avec les tiroirs 205
3.4 La fenêtre MainMenu.nib après ajout d'une fenêtre tiroir 206
3.5 Le content view d'un objet drawer (tiroir) 207
3.6 Un split view contenant un outline view et un table view 219
3.7 Un tab view avec trois onglets 222
3.8 Le compteur permettant de "switcher" d'un onglet à l'autre . . 226
4.1 Un bouton 255
4.2 Un objet color well dans une fenêtre 272
4.3 Un objet combo box sans sa liste visible 274
4.4 Un objet combo box avec sa liste déroulée 275

4.5 Une fenêtre affichant une image dans un objet image view (extrait de l'application "Image") . . . 286
4.6 Un objet matrix avec trois boutons radios . . . 289
4.7 Un objet movie view (extrait de l'application "Talking Head") . 297
4.8 Un bouton popup . . . 302
4.9 Une barre de progression indéterminée . . . 306
4.10 L'indicateur de progression circulaire indéterminé . . . 307
4.11 Un champ sécurisé affichant des ronds . . . 312
4.12 Sliders horizontaux et verticaux, sans et avec graduation . . . 316
4.13 Un objet stepper . . . 321
4.14 Des objets text field utilisés comme étiquette et champ de saisie 326
5.1 Un browser view affichant une partie du système de fichiers . . 364
5.2 L'application "Table" . . . 379
5.3 L'application "To Do list" . . . 383
5.4 Un objet outline view . . . 393
5.5 Un objet table view dans Interface Builder . . . 404
6.1 Le panneau "Groups & Files" d'un projet Xcode . . . 459
8.1 Le menu "File" de la fenêtre Nib d'Interface Builder . . . 496
8.2 L'élément de menu "New" du menu "File" de la fenêtre Nib d'Interface Builder . . . 499
9.1 Le panel "Couleurs" . . . 515
9.2 Le panel "Polices" . . . 520
9.3 Le panel "Ouvrir" . . . 523
9.4 Le panel d'enregistrement . . . 530
9.5 Un message d'alerte affiché en tant que "feuille" par la commande Display Alert . . . 541
9.6 Un dialogue "feuille" affiché par la commande Display Dialog . 544
11.1 Un text view contenant du texte . . . 570
12.1 La fenêtre de l'application "Archive Maker" . . . 582
12.2 La fenêtre de l'application "Unit Converter" avec les choix des types de mesure . . . 583
12.3 L'application "Unit Converter" avec son tiroir ouvert . . . 584
12.4 L'application "Table Sort" avec la liste des individus . . . 585
12.5 La colonne "Zip" retriée suite au clic sur son en-tête . . . 585
12.6 Sélection du contenu de la colonne "City" avant modification . 585
12.7 La fenêtre de l'application "Save Panel" avec les différents choix possibles . . . 586
12.8 Le panel d'enregistrement attaché à la fenêtre . . . 586
12.9 Le panel d'enregistrement affiché dans une fenêtre à part . . . 587

Première partie

Introduction

Chapitre 1

À propos de ce guide

AppleScript Studio est un environnement de développement permettant de créer rapidement des applications Mac OS X, en exécutant des scripts AppleScript et en respectant les directives de la guideline Aqua. Il allie les caractéristiques d'AppleScript, de Xcode, d'Interface Builder et du framework Cocoa application.

Xcode est un gestionnaire de projets, cette application est apparue et ne fonctionne qu'avec Mac OS X 10.3. Si votre système est antérieur, vous devrez utiliser Project Builder qui remplit les mêmes fonctions.

Ce guide décrit la terminologie de scripting d'AppleScript Studio version 1.3, livrée avec la version 10.3 de Mac OS X (Panther). Par contre, il ne décrit pas en détails le langage AppleScript, celui-ci est expliqué en détails dans le guide "*AppleScript Language Guide*", pour plus d'informations, voir la section "Autres documentations" (page 5).

Le chapitre "Les bases de la terminologie" (page 7) fournit certaines informations fondamentales sur la terminologie de scripting d'AppleScript Studio.

La terminologie des suites d'AppleScript Studio est décrite, par la suite, dans les parties suivantes :

Application Suite 27
Container View Suite 195
Control View Suite 251
Data View Suite 361
Document Suite 455

Drag and Drop Suite 475
Menu Suite 493
Panel Suite 511
Plugin Suite 557
Text View Suite 567

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) liste les différentes constantes mises à la disposition de toutes les suites.

Ce que ce guide contient

Ce guide décrit la terminologie fournie par AppleScript Studio version 1.3, disponible depuis la version 10.3 de Mac OS X (Panther). Il liste chaque terme de la terminologie d'AppleScript Studio, et indique ceux qui ne sont pas supportés par les versions antérieures. Pour plus d'informations sur les versions d'AppleScript Studio, voir "Information sur les versions" (page 8).

Ce guide tente aussi de répertorier les différences de terminologie entre les versions 1.0, 1.1 et 1.2 d'AppleScript Studio, en fournissant une section "Version" pour la terminologie ajoutée ou modifiée depuis la version 1.0. Si une classe, une commande, un Event ou une énumeration n'a pas de section "Version", cela signifie que cette terminologie est disponible et inchangée depuis la version 1.0 d'AppleScript Studio. Voir aussi "Information sur les versions" (page 8).

La plupart des classes utilisées dans les scripts AppleScript Studio (comme les classes application (page 29), view (page 229), button (page 254) et autres) sont construites respectivement à partir des classes Cocoa suivantes (NSApplication, NSDocument, NSButton, etc...), bien qu'elles ne reprennent pas toutes les fonctionnalités possibles de chaque classe Cocoa. Ce guide fournit les références à la documentation de chaque classe Cocoa correspondante comme une source supplémentaire d'informations sur la terminologie sous-jacente.

Pour plus d'informations sur l'organisation de la terminologie dans ce guide, voir "Comment sont organisées les suites ?" (page 12).

Ce que ce guide ne contient pas

Ce guide ne décrit pas en détails le langage AppleScript. Un autre guide le fait très bien, il s'agit du guide "*AppleScript Language Guide*".

Si vous ne connaissez pas encore le fonctionnement des scripts et les terminologies de scripting, vous trouverez dans le guide "*Inside Mac OS X : Building Applications With AppleScript Studio*" toute la documentation nécessaire.

Bien que ce guide fournisse de l'aide sur le fonctionnement de Xcode et Interface Builder pour créer des applications AppleScript Studio, il n'est pas une référence pour ces outils.

En décrivant les objets d'interface disponibles pour vos applications, ce guide mentionne certaines propriétés que vous pouvez régler dans Interface Builder, en plus des valeurs par défaut des propriétés des objets d'interface que vous instanciez. N'oubliez pas que la valeur de ces propriétés peut varier en fonction des réglages choisis dans l'application Interface Builder et de l'interaction des objets entre eux.

Autres documentations

Cette section liste différentes sources de documentation sur les composants d'AppleScript Studio. La documentation technique est disponible en ligne sur le site d'Apple consacré aux développeurs : `http://developer.apple.com/documentation/`. Si vous avez installé les outils développeurs, cette même documentation est disponible dans le dossier `/developer/Documentation` sur votre disque dur. La documentation est déclinée sous différents formats : HTML ou PDF, ou les deux. Elle peut être examinée, soit avec un navigateur internet, par exemple Safari ou Camino, soit dans la fenêtre Documentation de Xcode (disponible dans le menu "Help").

La documentation et les outils étant régulièrement mis à jour, si vous souhaitez être averti vous pouvez vous abonner au programme Online de mise à jour ADC et recevoir gratuitement la newsletter hebdomadaire par courriel. Voir la page `http://developer.apple.com/membership/` pour plus de détails sur ce programme.

Vous trouverez ci-dessous la liste des documentations utiles pour AppleScript Studio :

- *Building Applications With AppleScript Studio* introduit les ca-

ractéristiques clés d'AppleScript Studio et fournit des tutoriels détaillés sur la création d'une application AppleScript Studio.

- *AppleScript Language Guide* est le guide de base du langage AppleScript, en quelque sorte le mode d'emploi.
- *Technical Note TN2065 : do shell script in AppleScript* peut vous être utile si vous comptez utiliser des scripts Shell dans vos scripts AppleScript. Vous la trouverez à cette adresse : `http://developer.apple.com/technotes/tn2002/tn2065.html`.
- *Aqua Human Interface Guidelines* donne des recommandations sur l'utilisation des éléments d'interface (comme les fenêtres, les boutons, etc) et sur leur positionnement dans l'interface Aqua.
- la documentation sur Xcode et Interface Builder, disponible dans la partie "Tools Documentation".
- la documentation sur Cocoa, disponible dans la partie "Cocoa Documentation".

Chapitre 2

Les bases de la terminologie

La terminologie décrite dans ce guide permet aux applications AppleScript Studio de fonctionner avec les objets d'interface Cocoa dans les scripts. Ce chapitre fournit des informations importantes sur le fonctionnement de la terminologie de scripting d'AppleScript Studio. Il contient les sections suivantes :

Information sur les versions 8
Construire des applications AppleScript Studio 9
Les sources de la terminologie d'AppleScript Studio . . 11
Les suites 12
Comment sont organisées les suites ? 12
La terminologie fournie par le framework Cocoa Application 13
La terminologie fournie par AppleScript Studio 14
Les formes-clés standards 15
Les conventions d'appellation des méthodes et des gestionnaires 16
Accéder aux propriétés et aux éléments 17
Les paramètres des gestionnaires d'Events 19
Connecter les gestionnaires d'Events gérant le clavier et la souris 20
Les messages d'erreur de scripting 20
Utiliser les exemples de scripts 21
"Panels" contre "Dialogs" et "Windows" 21
Un mot sur Unicode Text 22

Information sur les versions

Pour construire des applications AppleScript Studio, vous devrez avoir installé une version des Developer Tools, incluant AppleScript Studio, adaptée à la version de votre système Mac OS X. Pour pouvoir faire tourner une application AppleScript Studio, votre Mac devra avoir le runtime AppleScript Studio requis par l'application. Le runtime AppleScript Studio est disponible si `AppleScriptKit.framework` est présent dans `/System/Library/Frameworks`.

AppleScript Studio essaie de faire en sorte que :

- une application créée et construite avec une ancienne version d'AppleScript Studio puisse fonctionner avec une version plus récente du runtime.
- une application créée et construite avec une version récente d'AppleScript Studio puisse fonctionner avec une version plus ancienne du runtime, toutefois si cette application n'utilise pas les nouvelles caractéristiques introduites par les versions plus récentes.

Par exemple, une application construite avec AppleScript Studio 1.1 et utilisant des caractéristiques introduites par cette version, requiert le runtime 1.1. Toutefois, la même application mais n'utilisant pas les nouvelles caractéristiques d'AppleScript Studio 1.1, pourra fonctionner avec le runtime 1.0. Et une application construite avec AppleScript Studio version 1.0 pourra fonctionner avec n'importe quel runtime, à travers la version 1.3.

Important
Une application créée avec AppleScript Studio version 1.2 ne fonctionnera pas en général avec les versions plus anciennes du runtime, même si elle n'utilise pas les nouvelles caractéristiques de la version 1.2. Cela a été corrigé dans la version 1.2.1 d'AppleScript Studio.

Le tableau 1.1 (page 9) liste pour chaque version d'AppleScript Studio, l'environnement de développement correspondant et le système Mac OS X minimum pour faire fonctionner une application AppleScript Studio.

Si vous voulez savoir comment une application peut déterminer si la version requise d'AppleScript Studio est présente ou non, voir la section "Exemples" du gestionnaire will finish launching (page 161).

Avec la version d'Interface Builder fournie avec Mac OS X version 10.2, vous aurez accès à la préférence Nib File Compatibility présente dans le panel "General" de la fenêtre "Preferences" d'Interface Builder. Vous pourrez alors sélectionner le niveau de compatibilité de votre application en optant pour

Version	Dispo. dans :	Supporté par :
1.0	Developer Tools Décembre 2001	Developer Tools CD (avec AppleScript 1.8.2), ou Mac OS X version 10.1.2 (avec AppleScript 1.8.3 ou supérieur)
1.1	Developer Tools Avril 2002	Developer Tools CD (avec AppleScript 1.8.2 ou supérieur)
1.2	Developer Tools Mac OS X 10.2	Mac OS X 10.2 et supérieur (avec AppleScript 1.9.0 ou supérieur)
1.2.1	Developer Tools Décembre 2002	Developer Tools CD (avec AppleScript 1.9.1) ou Mac OS X version 10.2.3 (avec AppleScript 1.9.1 ou supérieur)
1.3	Tools Xcode Mac OS X 10.3	Mac OS X 10.3 et supérieur (avec AppleScript 1.9.2 ou supérieur)

TAB. 1.1 - *Disponibilité de l'environnement de développement et du runtime d'AppleScript Studio*

un des choix suivants (n'oubliez pas de redémarrer Interface Builder pour que le changement soit effectivement pris en compte) :

Pre-10.2 format : vos applications fonctionneront avec des versions plus anciennes de Mac OS X, mais n'auront pas accès aux nouvelles caractéristiques (comme par exemple l'objet progress indicator (page 306) ou l'apparence métal brossé des fenêtres)

10.2 and later format : autorise l'accès aux nouvelles caractéristiques, mais ne garantit pas le bon fonctionnement avec les versions précédentes de Mac OS X

Both Formats : autorise l'accès aux nouvelles caractéristiques mais fonctionnera aussi avec les versions précédentes de Mac OS X (bien que sans les nouvelles caractéristiques)

Construire des applications AppleScript Studio

Vous utiliserez Xcode, distribué avec les Developer Tools, pour créer et construire des applications AppleScript Studio, en utilisant un des trois modèles suivants :

AppleScript Application : utilisé pour les applications qui n'ont pas besoin de stocker des données dans des documents

AppleScript Document-based Application : utilisé pour les applications qui créent et gèrent de multiples documents

AppleScript Droplet : utilisé pour la création de Droplets — des scripts-applications qui se déclenchent lorsque vous faites glissez l'icone d'un fichier ou d'un dossier sur leur icone dans le Finder

Vous utiliserez l'application Interface Builder, aussi distribuée avec les Developer Tools, pour créer des interfaces graphiques aux applications AppleScript Studio. Interface Builder fournit un support actif de l'interface utilisateur Aqua, comprenant un système de guides pour la mise en place des objets d'interface en accord avec la guideline Aqua.

Vous trouverez dans le guide "*Aqua Human Interface Guidelines*" des illustrations de fenêtres, menus, contrôles et autres éléments d'interface, ainsi que des recommandations sur leur utilisation et leur positionnement. S'il vous plaît, utilisez ce guide pour créer des applications qui tirent complètement partie de l'interface Aqua. La section "Autres documentations" (page 5) explique comment accéder à ce guide, ainsi qu'à la documentation sur Xcode et Interface Builder.

Toutes les applications AppleScript Studio sont des applications Cocoa, construites à partir du framework Cocoa application. En tant que telles, elles sont un mélange de codes Cocoa et de scripts AppleScript, mais vous pouvez écrire de robustes applications ne requérant pas de codes Cocoa personnels supplémentaires.

Certaines possibilités de Cocoa ne sont généralement pas disponibles dans les classes AppleScript Studio. Lorsque cela se produit, vous avez différentes solutions pour y remédier :

- vous pouvez utiliser la commande call method (page 94) pour appeler directement les méthodes des classes Cocoa.
- vous pouvez écrire votre propre code Cocoa comme faisant partie de votre application, et y accéder par l'intermédiaire de la commande Call Method. En fait, votre application peut accéder aux langages C, C++, Objective-C, Objective-C++ et Java (soit directement, soit par l'intermédiaire du "Java Bridge" de Mac OS X), comme le montre l'application "Multi-Language" distribuée avec AppleScript Studio.
- dans certains cas, les caractéristiques pourront être disponibles dans une future version d'AppleScript Studio.

Les sources de la terminologie d'AppleScript Studio

AppleScript Studio permet d'écrire des scripts qui contrôlent plusieurs applications, y compris des composants de Mac OS lui-même. Plusieurs outils existent pour l'écriture des scripts, comme l'application Éditeur de Scripts (distribuée avec Mac OS) ou d'autres produits de tierce-partie, comme Smile. La puissance des scripts vient d'abord de la terminologie de scripting fournie par les applications et le système Mac OS, pas du petit nombre de termes qui sont propres au langage AppleScript. Pour tirer le meilleur parti des capacités disponibles, vous devez savoir quelle terminologie employée dans vos scripts.

Les scripts des applications AppleScript Studio ont accès à la terminologie de base accessible par tous les scripts, comprenant :

- la terminologie fournie par le langage AppleScript
- la terminologie des composants pilotables du système Mac OS
- la terminologie de n'importe quel complément de pilotage, qu'il soit estampillé Apple ou autres (un complément de pilotage est du code, placé dans le dossier `SystemAdditions` des domaines System, Library, User et Network , fournissant des commandes ou des coercitions supplémentaires aux scripts installés sur le même ordinateur)
- la terminologie de toute application scriptable (Carbon ou Cocoa)

Les applications AppleScript Studio ont aussi accès à :

- leur propre terminologie en tant qu'application Cocoa (toutes les applications Cocoa dont le support du scripting est actif obtienne l'accès à la terminologie par défaut ; cette terminologie est décrite plus en détails dans "Les suites" (page 12))
- la terminologie définie par AppleScript Studio lui-même dans le framework `AppleScriptKit` (un framework est une sorte de bundle (page 38), ou de répertoire dans le fichier système, regroupant les ressources dont le programme a besoin)
- la terminologie des "Suites" (décrites dans "Les suites" (page 12)) de n'importe quel framework scriptable utilisé par l'application, et de n'importe quel bundle scriptable qu'il charge

Note : Les termes de ces diverses terminologies peuvent parfois créer des conflits, lesquels peuvent provoquer des erreurs d'interprétation lors de l'exécution des scripts.

Les suites

Le framework Cocoa propose ses informations de scripting sous forme de suites, lesquelles sont accessibles à toute application l'utilisant. Les applications peuvent aussi définir des suites supplémentaires. Une **suite** se compose au minimum d'une "suite definition" et d'une "suite terminology", contenues dans des fichiers séparés. Une "**suite definition**" décrit les objets scriptables avec des termes représentant leurs attributs, leurs relations et les commandes supportées. Ces informations sont stockées sous forme de paires "clé-valeur" dans une propriété liste. Une **propriété liste** est une représentation structurée et textuelle de données, généralement enregistrée au format XML (Extensible Markup Language).

Une "**suite terminology**" fournit la terminologie AppleScript - les termes ou phrases en langue anglaise utilisables dans les scripts - correspondant aux descriptions de classes ou de commandes d'une "suite definition". Les "suites terminology" sont aussi stockées sous forme de propriétés listes. Les frameworks et les applications placent les fichiers de terminologie dans un répertoire de ressources localisées, nommé `English.lproj`. L'anglais est l'unique dialecte supporté par AppleScript.

Note : Les applications Carbon stockent la terminologie de scripting dans un format différent - dans une ressource 'aete' (terminologie Apple Event).

Comment sont organisées les suites ?

Lorsque c'est possible, ce guide fournit les liens renvoyant aux définitions des classes, des commandes, des Events, des propriétés, des éléments ou des constantes. Pour chaque suite d'AppleScript Studio, ce guide fournit les informations suivantes :

- un bref aperçu de la suite
- la terminologie de chaque classe
  - forme plurielle, classe parente et les classes Cocoa sur lesquelles se base la classe

Note : Les classes qui commencent par "NS" (comme NSButton) sont définies dans le framework Cocoa et renvoient à la documentation Cocoa. Les classes qui commencent par "ASK" (comme ASKDataCell) sont définies dans le propre framework d'AppleScript Studio, le framework `AppleScriptKit`.

  - propriétés, éléments, commandes et Events supportés
  - version (si nouveauté ou modification depuis la version 1.0)
- la terminologie de chaque commande et Event (s'il y en a)
  - résumé, syntaxe, description des paramètres et des valeurs retournées (s'il y en a)
  - exemples
  - discussion (s'il y en a)
  - version (si nouveauté ou modification depuis la version 1.0)
- la terminologie de chaque Énumeration (toutes les valeurs de Énumeration, lesquelles sont disponibles pour chaque suite, sont décrites dans "Énumérations" (page 173) de "Application Suite" (page 27))
  - une description de l'énumération
  - une description de chaque constante de l'énumération
  - version (si nouveauté ou modification depuis la version 1.0)

La terminologie fournie par le framework Cocoa Application

Les applications AppleScript Studio peuvent tirer profit de la terminologie de Cocoa accessible depuis deux suites par défaut :

- La Standard Suite
  - définit les classes de base, comprenant Application, Document et Window (bien qu'AppleScript Studio définit ses propres versions pour ces classes dans "Application Suite" (page 27)).
  - définit la terminologie des Events de base, comprenant Get, Set, Count, Delete, Print, Quit et autres. Dans les applications Cocoa dont le support du scripting est actif, les objets peuvent supporter certaines commandes importantes, comme Get et Set, avec un peu ou pas du tout de code développeur supplémentaire.
- La Text Suite définit les classes fonctionnant avec le texte, comme les classes Character, Paragraph, Word et Text. Pour plus d'informations sur le travail avec le texte, voir la section "Exemples" de la classe text view (page 569).

Important
Le framework Cocoa application fournit des suites par défaut pour supporter le scripting des applications Cocoa. Bien que les applications AppleScript Studio aient accès à cette terminologie, elles utiliseront en priorité la terminologie décrite dans la section suivante.

La terminologie fournie par AppleScript Studio

AppleScript Studio définit ses propres suites, lesquelles s'ajoutent à celles déjà définies par Cocoa. Ces suites fournissent une terminologie supplémentaire utilisable sur les objets des scripts, basée sur la plupart des classes du framework Cocoa application. Cette terminologie est définie dans plusieurs fichiers suite définition et suite terminologie dans le propre framework d'AppleScript Studio, le framework `AppleScriptKit`. Chaque suite peut comporter des définitions de classes (lesquelles comprennent les éléments et les propriétés), les définitions de commandes et d'Events (lesquelles ont une syntaxe associée) et des énumérations (ou des constantes prédéfinies).

Note : AppleScript Studio fait une distinction entre une **commande** (un mot ou une phrase qu'un script peut envoyer pour provoquer la modification d'un objet) et un **Event** (une action, généralement générée par l'utilisateur, qui provoque l'exécution du gestionnaire de l'objet approprié).
C'est à dire, que les scripts peuvent envoyer des commandes aux objets, tandis que les Events, souvent le résultat d'une action de l'utilisateur, génèrent les appels aux gestionnaires des scripts.

Chacun des chapitres suivants décrit la terminologie des suites d'AppleScript Studio :

Application Suite 27
Container View Suite 195
Control View Suite 251
Data View Suite 361
Document Suite 455
Drag and Drop Suite 475
Menu Suite 493
Panel Suite 511

Plugin Suite 557

Text View Suite 567

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) liste les différentes constantes mises à la disposition de toutes les suites.

Les formes-clés standards

Pour accéder à la propriété ou à l'élément d'un objet, un script peut les spécifier par n'importe quelle forme-clé (key form) supportée. Les formes-clés indiquent simplement comment les données doivent être interprétées. Par exemple, si une propriété supporte les formes-clés spécifiant une position relative, un script peut utiliser une instruction telle que celle- ci :

```
set myWindow to the third window
```

Les objets des applications AppleScript Studio sont basés sur les définitions des classes Cocoa, et le support du scripting d'AppleScript Studio permet à beaucoup de classes de convenablement supporter une grande variété de formes-clés. Avec comme résultat que la plupart des objets des applications AppleScript Studio supportent les formes-clés suivantes :

- Name
  `tell drawer "myDrawer" to open drawer on top edge`
- Absolute Position (index numérique)
  `text field 1 of window 1`
- Relative Position (avant / après)
  `window in front of window 3`
- Range (une sélection d'éléments)
  `name of window 1 through 3`
- Satisfaisant à un test
  `the first window whose name is "mainWindow"`
- ID (unique)
  `set windowID to ID of window 1`

Voir aussi "Accéder aux propriétés et aux éléments" (page 17).

Les conventions d'appellation des méthodes et des gestionnaires

Le framework Cocoa application fournit une convention grammaticale (basée sur le langage humain) permettant de déterminer à quel moment, lors de l'exécution du code, une méthode sera appelée. Cette convention, laquelle est reprise dans la terminologie des gestionnaires d'Events d'AppleScript Studio, insère "should", "will" et "did" dans le nom des méthodes. Le tableau 1.2 décrit la signification de ces termes. Notez que pour indiquer une opération déjà achevée, AppleScript Studio utilise le temps passé, plutôt que le terme "did".

Expression Cocoa	Explication	Exemples
`should`	Demande si une opération doit avoir lieu. Vous pouvez annuler l'opération en retournant `false`.	`should open` `should close`

`will`	Une opération est sur le point d'avoir lieu. Vous pouvez la préparer, mais vous ne pouvez pas l'empêcher.	`will resize`
		`will hide`
		`will quit`

`did`	Une opération s'est achevée. Vous pouvez exécuter des actions en réponse. AppleScript Studio utilise le temps passé, plutôt que le terme "did".	`activated`
		`launched`
		`miniaturized`
		`zoomed`

TAB. 1.2 - *Conventions d'appellation de Cocoa et d'AppleScript Studio*

Aussi, par exemple, vous pouvez ajouter un gestionnaire should close (page 151) à un objet window (page 76). Lorsque le gestionnaire est appelé, il peut déterminer si l'utilisateur a exécuté certaines tâches essentielles — dans le cas contraire, il peut retourner `false` et refuser d'autoriser la fermeture de la fenêtre. Un gestionnaire will close (page 161) ne peut pas annuler l'opération de fermeture, mais il peut exécuter n'importe quelles tâches pour préparer cette opération. Finalement, un gestionnaire closed (page 129) peut exécuter n'importe quelles tâches requises après la fermeture.

Au démarrage de l'application, les gestionnaires connectés à l'objet application (page 29) sont appelés dans cet ordre :

1. will finish launching (page 161)
2. awake from nib (page 123)

> **Important** : Si vous travaillez avec la version d'Interface Builder livrée avec Mac OS X version 10.2, voir "Information sur les versions" (page 8) pour des informations sur le réglage de la préférence "Nib File Compatibility".

3. launched (page 135)
4. will become active (page 160)
5. activated (page 123)
6. idle (page 132)

Accéder aux propriétés et aux éléments

Alors que les éléments ont généralement un nom au singulier, comme "document", "window" ou "pasteboard", vous utiliserez la forme plurielle pour accéder à certains éléments de la classe application (page 29). Par exemple, vous pourrez utiliser le script suivant dans l'Éditeur de Scripts pour obtenir la liste des fenêtres d'une application AppleScript Studio (et vous pouvez utiliser une terminologie identique à l'intérieur d'une application AppleScript Studio, mais vous n'aurez pas besoin du bloc `tell application`) :

```
tell application "MyApp"
  set windowList to windows
end
```

Une classe peut lister les propriétés auxquelles vous ne pouvez pas accéder dans une application particulière ou à un moment particulier. Par exemple, la classe application (page 29) propose une propriété *key window* qui identifie la fenêtre comme étant la principale cible du clavier. Mais si une application n'a pas de fenêtre ouverte, il n'y aura pas de fenêtre-clé, et essayer d'accéder à la propriété *key window* retournera une valeur zéro. Pour plus d'informations sur la fenêtre-clé, voir les descriptions des propriétés *key* et *main* de la classe window (page 76).

Depuis la version 1.2 d'AppleScript Studio, vous pouvez utiliser la propriété *properties* pour accéder aux propriétés de la plupart des objets d'AppleScript Studio. Cette caractéristique est maintenant disponible grâce à la nouvelle version de Cocoa distribuée avec Mac OS X version 10.2. Le script suivant montre comment afficher les propriétés d'un bouton dans une application AppleScript Studio :

```
tell application "simple"
```

```
  properties of button 1 of window 1
end tell
```

Le résultat de ce script pourra être quelque chose comme ce qui suit, où `missing value` indique qu'une valeur pourrait ne pas être obtenue pour une certaine propriété :

```
{visible:true, content:false, double value:0.0, ignores
multiple clicks:false, title:"Button", image:missing
value, name:missing value, bordered:true, enabled:true,
window:window id 1 of application "TestingControls", allows
mixed state:false, float value:0.0, image position:no image,
string value:"0", id:2, flipped:true, roll over:false,
opaque:false, alternate image:missing value, bounds:{47,
45, 131, 77}, bounds rotation:0.0, tag:0, class:button,
alignment:center text alignment, enclosing scroll view:missing
value, can draw:true, menu:missing value, target:missing
value, bezel style:rounded bezel, alternate title:"",
needs display:false, current cell:item id 3 of application
"TestingControls", auto resizes:false, key equivalent
modifier:0, position:{47, 45}, formatter:missing value,
continuous:false, size:{84, 32}, cell:item id 3 of application
"TestingControls", state:0, button type:momentary push in
button, font:item id 4 of application "TestingControls",
sound:missing value, integer value:0, tool tip:missing
value, super view:content view of window id 1 of application
"TestingControls", key equivalent:"", current editor:missing
value, transparent:false, visible rect:{0, 0, 84, 32},
contents:false}
```

Si votre application risque d'accéder à une propriété pouvant ne pas avoir de valeur, vous devrez mettre les instructions utilisant cette propriété dans un gestionnaire d'erreurs (`try...on error...end try`). Pour un exemple, voir la section "Exemples" de la commande path for (page 112).

De même, suivant le contexte, un objet pourrait avoir pour sa classe aucun ou plusieurs éléments listés. Si vous essayez d'accéder à un élément pour lequel il n'y a pas d'instances, vous obtiendrez une liste vide. Vous n'aurez pas besoin d'un gestionnaire d'erreurs puisqu'un résultat est retourné, mais suivant la logique de votre script, vous pourriez avoir besoin de vérifier que la liste n'est pas vide.

Important : Ce guide indique parfois que certaines propriétés peuvent être réglées dans Interface Builder et qu'elles ont des valeurs par défaut, mais ce guide n'essaie pas d'être un guide exhaustif sur Interface Builder, pour cela il est préférable de consulter l'aide propre à cette application.

Les paramètres des gestionnaires d'Events

Dans une application AppleScript Studio, vous utiliserez Interface Builder pour connecter les objets de l'application avec les gestionnaires d'Events de vos scripts. Par exemple, si vous connectez un gestionnaire clicked (page 348) à un bouton, lorsque l'utilisateur cliquera sur ce bouton, le gestionnaire sera appelé dans le script de votre application.

Lorsque vous connecterez pour la première fois, un gestionnaire dans Interface Builder, celui-ci installera automatiquement un gestionnaire vierge dans le script désigné. Le listing 1.1 montre le gestionnaire vierge d'un gestionnaire clicked (page 348). Les gestionnaires vierges sont toujours ajoutés avec leur syntaxe complète, le gestionnaire ici n'a qu'un seul paramètre, `theObject`. Tous les gestionnaires vierges comportent un paramètre `theObject` (lequel sera presque toujours une référence à l'objet pour lequel le gestionnaire est appelé), mais certains gestionnaires ont aussi des paramètres supplémentaires.

```
on clicked theObject
  (* Add your script here. *)
end clicked
```

LIST. 1.1 - *Un gestionnaire Clicked vierge*

La définition de la syntaxe de chaque gestionnaire d'Events montre la syntaxe complète utilisée pour construire le gestionnaire vierge. Une fois qu'un gestionnaire vierge a été inséré dans le script de l'application, vous êtes libre de modifier les noms de ces paramètres ou de supprimer (ou d'ignorer) tout paramètre déclaré comme facultatif dans la définition de la syntaxe.

Une des manières de faciliter la documentation des scripts est, lorsque c'est possible, de modifier le nom du paramètre `theObject` afin qu'il soit représentatif de l'objet auquel il se réfère. Par exemple, si un gestionnaire clicked (page 348) est uniquement appelé en réponse au clic sur un bouton Search, vous pouvez modifier la déclaration du gestionnaire d'Events en `on clicked searchButton`. Modifier le nom d'un paramètre n'a aucun effet

sur le fonctionnement du gestionnaire.

Si le gestionnaire peut être appelé par un ou plusieurs objets (comme une série de boutons présents sur un objet tab view (page 221)), vous pouvez à la fois modifier le nom du paramètre et rajouter un commentaire pour clarifier son utilisation :

```
on clicked importButton
  (* This handler handles buttons on the Import pane *)
  -- your script statements here
end clicked
```

Connecter les gestionnaires d'Events gérant le clavier et la souris

AppleScript Studio fournit un certain nombre de gestionnaires pour gérer le clavier et la souris, comme keyboard down (page 133), keyboard up (page 134), mouse down (page 137), mouse up (page 141), mouse dragged (page 138), etc. Pour certains objets, particulièrement les objets application (page 29), les gestionnaires d'Events du clavier et de la souris qui y seront connectés peuvent ne jamais être appelés, car ils sont d'abord gérés par d'autres objets avant l'objet Application. Si votre application a réellement besoin de traiter avec ces gestionnaires, tenez compte de leurs connexions avec les objets présents dans l'interface qui héritent de la classe control (page 280), comme les objets button (page 254), slider (page 315), stepper (page 320), ou text field (page 325).

Les messages d'erreur de scripting

Les applications AppleScript Studio sont des applications Cocoa, aussi lorsque votre application retourne un message d'erreur signifiant une anomalie de scripting, ce message est généré par le support du scripting de Cocoa. Le tableau 1.3 (page 23) liste les erreurs de scripting jusqu'à la version 10.2 de Mac OS X, ainsi que les numéros d'erreur correspondant.

Ces messages peuvent ne pas fournir tous les détails que vous aimeriez pour le débogage de votre application, aussi d'autres options sont disponibles, comme le système de débogage de Xcode. Avec ce système vous pourrez régler des points d'arrêt, examiner des variables ou exécuter pas à pas vos scripts. Vous pouvez aussi utiliser la commande log (page 111)

d'AppleScript Studio pour récupérer les valeurs et les messages produits lors de l'exécution de votre application.

Utiliser les exemples de scripts

Ce guide contient de nombreux exemples de scripts et de gestionnaires d'Events, complets ou partiels. Dans certains cas, les longues instructions (sur plusieurs lignes) contiennent des retours-chariots afin de faciliter leur lecture. Si vous copiez-collez des exemples de scripts depuis ce guide dans les scripts de vos applications AppleScript Studio, ou dans les scripts que vous utiliserez avec l'Éditeur de Scripts ou une autre application, vous devrez peut-être supprimer ces retours-chariots afin de pouvoir les compiler.

"Panels" contre "Dialogs" et "Windows"

Pour des raisons historiques, le framework Cocoa application utilise le terme "panel" dans de nombreux cas où "*Aqua Human Interface Guidelines*", l'arbitre final pour les choix d'interface Mac OS X, utiliserait plutôt les termes "dialog" ou "window". L'utilisation de "panel" est même ancrée dans le nom des classes Cocoa, comme NSPanel, NSFontPanel, NSOpenPanel, etc. La terminologie d'AppleScript Studio, basée sur les classes d'interface Cocoa, a elle-aussi adopté ce système de noms et fournit "Panel Suite" (page 511), laquelle inclut les classes color-panel (page 514), font-panel (page 520), open-panel (page 522), et autres classes de même nature.

Connue cette histoire, il n'est guère possible pour ce guide d'être complètement cohérent lors de l'utilisation de ces termes. Toutefois, la liste suivante montre comment certaines utilisations de "panel" correspondent aux termes utilisés dans le guide "*Aqua Human Interface Guidelines*". Voir "Autres documentations" (page 5) pour savoir comment obtenir le guide "*Aqua Human Interface Guidelines*".

- panel (page 526) (basé sur la classe NSPanel) est un type spécial de fenêtres ayant normalement dans une application une fonction auxiliaire, comme fournir le "status" à l'utilisateur. Le guide "*Aqua Human Interface Guidelines*" se réfèrera à ces types de fenêtres avec "dialogs" ou "utility windows".
- color-panel (page 514) (basé sur la classe NSColorPanel) fournit une interface graphique standard permettant de sélectionner une couleur dans une application. Le guide Aqua l'appellera "Colors windows".

- font-panel (page 520) (basé sur la classe NSFontPanel) affiche la liste des polices disponibles, autorisant la prévisualisation et la sélection. Le guide Aqua l'appellera "Fonts window".
- open-panel (page 522) (basé sur la classe NSOpenPanel) fournit un mécanisme standard permettant à l'utilisateur de spécifier le nom du fichier à ouvrir. Le guide Aqua l'appellera "Open dialog".
- save-panel (page 529) (basé sur la classe NSSavePanel) autorise l'utilisateur à spécifier le répertoire et le nom sous lequel un fichier est enregistré. Le guide Aqua l'appellera "Save dialog".

Un dialogue peut être affiché comme une "feuille" ("sheet") (attaché à une fenêtre), dans ce cas c'est un document modal, et l'utilisateur pourra travailler avec les autres fenêtres de l'application sans avoir besoin de la fermer. Par exemple, vous pouvez utiliser le paramètre facultatif `attached to` de la commande display (page 537) pour afficher un dialogue open-panel (page 522) comme une "feuille".

Un mot sur Unicode Text

Lorsque vous obtenez un texte en retour d'AppleScript Studio, il sera presque toujours au format Unicode. Dans certains cas, vous pourrez avoir besoin de convertir ce texte en texte brut. Pour une discussion et un exemple sur la manière de convertir de l'Unicode en texte brut, voir la section "Discussion" de la classe default entry (page 45).

Dans certains cas, lorsque vous fournissez du texte à AppleScript Studio, comme lors de l'utilisation de la commande localized string (page 109), vous devrez vous assurer que vous travaillez bien avec le format Unicode. Voir la section "Exemples" de localized string (page 109) pour savoir comment spécifier le format UTF-8 dans votre projet d'application.

Constante (N°)	Description
`NSNoScriptError (0)`	Aucune erreur.
`NSReceiverEvaluationScriptError (1)`	L'objet ou les objets spécifiés à une commande par le paramètre direct pourraient ne pas être trouvés.
`NSKeySpecifierEvaluationScriptError (2)`	L'objet ou les objets spécifiés par une référence pourraient ne pas être trouvés.
`NSArgumentEvaluationScriptError (3)`	L'objet spécifié à une commande par un argument pourrait ne pas être trouvé.
`NSReceiversCantHandleCommandScriptError (4)`	L'objet ne supporte pas la commande qui lui a été envoyée.
`NSRequiredArgumentsMissingScriptError (5)`	Un ou plusieurs arguments requis sont manquants.
`NSArgumentsWrongScriptError (6)`	Un argument (ou plusieurs) est du mauvais type ou sinon invalide.
`NSUnknownKeyScriptError (7)`	Une erreur non-identifiée est survenue ; indique une erreur dans le support du scripting d'une application scriptable, ou d'AppleScript Studio (ou Cocoa) lui-même.
`NSInternalScriptError (8)`	Une erreur non-identifiée est survenue ; indique une erreur dans le support du scripting d'une application scriptable, ou d'AppleScript Studio (ou Cocoa) lui-même.
`NSOperationNotSupportedForKeyScriptError (9)`	L'opération demandée n'a pas supporté la référence spécifiée.
`NSCannotCreateScriptCommandError (10)`	L'application a reçu un Apple Event invalide ou non-reconnu.

TAB. 1.3 - *Les messages d'erreur de scripting de Cocoa*

Deuxième partie

Application Suite

Cette partie décrit la terminologie de la suite Application d'AppleScript Studio.

La suite Application donne sa propre version de certaines classes généralement utilisées par ou avec un objet application, comprenant les nouvelles versions de certaines classes définies dans la Standard Suite.

Les classes qui remplacent les classes de la Standard Suite sont les classes application (page 29) et window (page 76). La Standard Suite est décrite dans "La terminologie fournie par le framework Cocoa Application" (page 13).

La suite Application définit aussi la classe item (page 60), laquelle possède les propriétés *name* et *ID*, et la classe responder (page 68), laquelle hérite de la classe item (page 60) et sert comme super-classe pour les classes window (page 76), view (page 229) et control (page 280). Toutes les classes héritant de responder (page 68) peuvent répondre aux actions de l'utilisateur.

Pour fonctionner avec les nombreuses classes de haut niveau qu'elle contient, la suite Application définit un grand nombre d'Events et de commandes pour travailler avec les Events Application, Fenêtre, Souris, Clavier, etc.

Les classes, commandes, Events et énumérations de la suite Application sont décrits dans les chapitre suivants :

Classes 29
Commandes 93
Events 121
Énumérations 173

Chapitre 1

Classes

La suite Application définit les classes suivantes :

application 29
bundle 38
data 45
default entry 45
event 50
font 54
formatter 56
image 58
item 60
movie 63
pasteboard 64
responder 68
sound 70
user-defaults 72
window 76

`application`

Pluriel : `applications`
Hérite de : `responder` (page 68)
Classe Cocoa : NSApplication

Gère la boucle principale d'Events d'un objet application, ainsi que les ressources utilisées par tous les objets de l'application.

La fonction principale d'un objet application est de recevoir les Events et de les distribuer aux objets appropriés pouvant y répondre (typiquement les sous-classes view). Par exemple, tous les Events Clavier et Souris sont directement envoyés à l'objet window (page 76) associée à l'Event. Dans une application AppleScript Studio, ces Events peuvent être dispatchés aux scripts des gestionnaires d'Events que vous avez connectés dans Interface Builder. Voir la classe responder (page 68) pour plus d'informations sur la gestion par l'application des Events Clavier et Souris.

Toute application AppleScript Studio (et Cocoa) n'a exactement qu'une seule instance de l'objet application, créée automatiquement comme partie intégrante du projet d'application dans Xcode. L'illustration 2.1 montre l'instance File's Owner du fichier nib principal. Dans le fichier nib principal, File's Owner représente toujours NSApp, une constante globale qui référence l'objet NSApplication de l'application. L'objet application sert comme contrôleur principal de l'application. Vous attacherez les gestionnaires à l'objet application dans Interface Builder par l'intermédiaire de l'instance File's Owner. Pour plus d'informations sur les fichiers nib et File's Owner, voir la commande awake from nib (page 123).

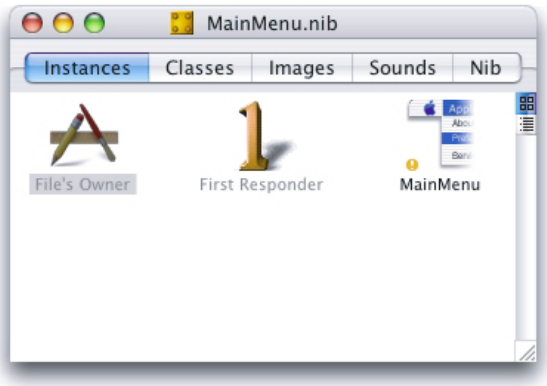

FIG. 2.1 - *L'instance File's Owner (représentant l'objet application) dans Interface Builder*

L'objet application fournit des instances partagées de color-panel (page 514), font-panel (page 520), open-panel (page 522) et save-panel (page 529) utilisables dans les scripts d'une application. L'objet application fournit aussi des éléments pour mémoriser des éléments comme les objets image (page 58), movie (page 63) et sound (page 70) avec l'application.

Propriétés des objets de la classe Application

En plus des propriétés héritées de responder (page 68), un objet application possède ces propriétés (consulter la section "Version" de cette classe pour connaître dans quelle version d'AppleScript Studio fut ajoutée une propriété particulière) :

active

Accès : lecture / écriture
Classe : *boolean*
L'application est-elle active ?

color panel

Accès : lecture / écriture
Classe : *color-panel* (page 514)
Le panel "Couleurs" de l'application ; créé automatiquement pour chaque application AppleScript Studio

font panel

Accès : lecture / écriture
Classe : *font-panel* (page 520)
Le panel "Polices" de l'application ; créé automatiquement pour chaque application AppleScript Studio

hidden

Accès : lecture / écriture
Classe : *boolean*
L'application est-elle cachée ? Régler la propriété *hidden* sur *true* règlera la propriété *visible* de tous les objets window (page 76) de l'application sur *false* ; l'inverse restaurera les réglages précédents de la propriété *visible* de toutes les fenêtres

icon image

Accès : lecture / écriture
Classe : *image* (page 58)
L'icone de l'application ; l'icone par défaut est un crayon, un pinceau et une règle placés de façon à représenter la lettre "A". Vous pouvez modifier l'icone en spécifiant un fichier icone dans Xcode : commencez par sélectionner la cible courante dans la liste Groups & Files, puis ouvrez la fenêtre Info (avec Cmd + I) et choisissez l'onglet Properties, là vous pouvez saisir un nom de fichier dans le champ Icon Field

key window

Accès : lecture uniquement
Classe : *window* (page 76)
La fenêtre clé courante (la cible courante des périphériques de saisie ; consulter les propriétés *key* et *main* de la classe window (page 76) pour plus d'informations)

main bundle

Accès : lecture / écriture
Classe : *bundle* (page 38)
Le bundle principal de l'application ; le bundle principal est l'emplacement par défaut de toutes les ressources de l'application, y compris les scripts compilés ; il est créé automatiquement par Xcode

main menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu principal de l'application ; les éléments de menus du menu principal représente généralement d'autres menus ; consulter la section "Exemples" de cette classe pour plus d'informations

main window

Accès : lecture uniquement
Classe : *window* (page 76)
La fenêtre principale de l'application ; la fenêtre principale est le centre d'action de l'activité de l'utilisateur ; consulter les propriétés *key* et *main* de la classe window (page 76) pour plus d'informations

name

Accès : lecture uniquement
Classe : *Unicode text*
Le nom de l'application

open panel

Accès : lecture / écriture
Classe : *open-panel* (page 522)
Le panel d'ouverture de l'application ; créé automatiquement pour chaque application AppleScript Studio

save panel

Accès : lecture / écriture

Classe : *save-panel* (page 529)
Le panel d'enregistrement de l'application ; créé automatiquement pour chaque application AppleScript Studio

services menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu "Services" de l'application ; les services permettent à une application de tirer avantages des caractéristiques fournies par d'autres applications, comme la vérification orthographique ou l'envoi par mail de la sélection en cours

user defaults

Accès : lecture / écriture
Classe : *user-defaults* (page 72)
Les couples de valeurs utilisateurs par défaut de l'application (voir aussi default entry (page 45))

version

Accès : lecture uniquement
Classe : *Unicode text*
La version de l'application, issue de la chaîne de caractères de la version courte ; par défaut, cette propriété vaut "0"

windows menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu "Fenêtre" de l'application ; par défaut, le menu "Fenêtre" mis en place par Interface Builder contient "Minimize" (Masquer la fenêtre) et "Bring All to Front Items" (Tout ramener au premier plan) ; dans une application active, il liste aussi toutes les fenêtres ouvertes de l'application

Éléments des objets de la classe Application

Un objet application peut contenir les éléments listés ci-dessous. Votre script peut, pour la plupart, les spécifier avec les formes-clés décrites dans "Les formes-clés standards" (page 15). Consulter la section "Version" de cette classe pour connaître les modifications et les ajouts survenus depuis la version 1.0 d'AppleScript Studio.

`data source` (page 386)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets data source de l'application

`document` (page 457)
Spécifier par : "Les formes-clés standards" (page 15)
Les documents de l'application

`drag info` (page 477)
Spécifier par : "Les formes-clés standards" (page 15)
À l'usage interne et exclusif d'AppleScript Studio

`event` (page 50)
Spécifier par : "Les formes-clés standards" (page 15)
L'Event en cours — le dernier Event extrait de la file d'attente des Events

`image` (page 58)
Spécifier par : "Les formes-clés standards" (page 15)
Les images de l'application ; par défaut, une application n'a accès qu'à une seule image, l'image de l'icone par défaut (décrite dans la section "Propriétés" plus haut) ; vous pouvez ajouter des images (y compris des images supplémentaires d'icones) à un projet dans Xcode ou Interface Builder, mais elles ne seront pas ajoutées aux éléments de l'application tant que vous ne les aurez pas chargées avec la commande load image (page 100)

`item` (page 60)
Spécifier par : "Les formes-clés standards" (page 15)
À l'usage interne et exclusif d'AppleScript Studio

`movie` (page 63)
Spécifier par : "Les formes-clés standards" (page 15)
Les films de l'application ; par défaut, une application n'a aucun film ; vous pouvez ajouter des films à un projet dans Xcode ou Interface Builder, mais ils ne seront pas ajoutés aux éléments de l'application tant que vous ne les aurez pas chargés avec la commande load movie (page 104)

`pasteboard` (page 64)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets pasteboard de l'application

`sound` (page 70)
Spécifier par : "Les formes-clés standards" (page 15)
Les sons de l'application ; par défaut, une application accède aux sons présentés dans l'illustration 2.6 (les sons disponibles peuvent différer suivant les systèmes) ; vous pouvez ajouter des sons à un projet dans Xcode ou Interface Builder, mais ils ne seront pas ajoutés aux éléments de l'application tant que vous ne les aurez pas chargés avec la commande load sound (page 107)

`window` (page 76)
Spécifier par : "Les formes-clés standards" (page 15)
Les fenêtres de l'application

Commandes supportées par les objets de la classe Application

Votre script peut envoyer les commandes suivantes à un objet application :

`display dialog` (page 543)
`path for` (page 112)
`quit` (de la Core Suite Cocoa)

Events supportés par les objets de la classe Application

Un objet application supporte les gestionnaires répondant aux Events suivants. Notez toutefois que les Events Clavier et Souris peuvent être gérés par d'autres objets, et cette multi-gestion peut provoquer des problèmes de communication entre ces gestionnaires et l'objet application, certains appels restant parfois sans réponse.

Application

`activated` (page 123)
`idle` (page 132)
`launched` (page 135)
`open` (de la Core Suite Cocoa)
`open untitled` (page 143)
`resigned active` (page 145)
`should open untitled` (page 153)
`should quit` (page 154)
`should quit after last window closed` (page 155)

`shown` (page 157)
`was hidden` (page 159)
`will become active` (page 160)
`will finish launching` (page 161)
`will hide` (page 163)
`will quit` (page 166)
`will resign active` (page 167)
`will show` (page 169)

Document

`document nib name` (page 130)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

Exemples

Le gestionnaire launched (page 135) suivant, connecté à l'objet application par l'intermédiaire de l'instance File's Owner dans Interface Builder, règlera la couleur du panel "Couleurs" de l'application sur le Rouge et affichera ce panel lors du lancement de l'application. Les instructions des scripts d'AppleScript Studio peuvent se référer aux propriétés de l'application sans explicitement cibler cet objet application.

```
on launched theObject
  set color of color panel to {65535,0,0}
  set visible of color panel to true
end launched
```

Vous pouvez utiliser le script suivant dans l'Éditeur de Scripts pour obtenir les titres des éléments de menu du menu principal de l'application "Drag Race", application distribuée avec AppleScript Studio. Ces mêmes instructions fonctionneront aussi à l'intérieur d'un script d'une application AppleScript Studio (bien que vous n'aurez pas besoin du bloc `tell application`).

```
tell application "Drag Race"
  title of menu items of main menu
  -- résultat : {"","File","Edit","Window","Help"}
end tell
```

Ce script peut également être plus précis pour obtenir des informations sur les éléments de menu du menu principal. Par exemple, la ligne suivante (si elle était insérée dans le bloc Tell précédent) obtiendrait le titre d'un élément de menu du menu "Fichier" :

```
title of menu item 10 of sub menu of menu item 2 of main menu
-- résultat : "Mise en page"
```

Il peut être plus facile d'utiliser l'application Éditeur de Scripts (ou une application de tierce partie) pour déterminer la syntaxe correcte identifiant les objets d'une application AppleScript Studio, comme montré ici, spécialement lorsque vous n'avez pas donné de noms AppleScript à ces objets dans Interface Builder. Maintenant supposons que les éléments de menu ont bien été nommés, alors vous pourrez utiliser une instruction telle que celle-ci :

```
title of menu item "page setup" of sub menu item "file" of main menu
```

Version

Les propriétés suivantes furent ajoutées à l'objet application dans la version 1.1 d'AppleScript Studio : *color panel, font panel, open panel, save panel, user defaults*.

Les éléments suivants furent ajoutés à l'objet application dans la version 1.1 d'AppleScript Studio : `data source, item, sound`.

Les éléments suivants furent ajoutés à l'objet application dans la version 1.2 d'AppleScript Studio : `drag info, pasteboard`.

bundle

Pluriel : `bundles`
Hérite de : `item` (page 60)
Classe Cocoa : NSBundle

Représente un emplacement dans le fichier système regroupant le code et les ressources pouvant être utilisés dans un programme.

Chaque objet application (page 29) a une propriété *main bundle* représentant le bundle principal de cette application. Bien que ce soit rare dans les applications AppleScript Studio, une application peut contenir des bundles supplémentaires.

Un objet bundle représente le répertoire où des ressources liées — y compris le code exécutable — sont stockées. Un bundle peut trouver les ressources demandées dans le répertoire et peut dynamiquement charger le code exécutable (bien que cela ne sera pas nécessaire pour la plupart des applications AppleScript Studio). Un objet bundle a des propriétés qui spécifient son emplacement dans le fichier système, ainsi que l'emplacement de certains de ses éléments. Vous pouvez aussi utiliser la commande path for (page 112) pour obtenir le chemin des éléments du bundle de l'application.

Un bundle peut contenir des images, des sons, des chaînes de caractères localisées et des plugs-ins. Il contient aussi le fichier `Info.plist` de l'application, lequel spécifie diverses informations sur l'application et le bundle pouvant être utilisées lors de son fonctionnement, y compris les types de documents, le numéro de version et les informations de copyright. Pour un exemple sur la vérification de la présence de la version minimum du runtime d'AppleScript Studio exigée par une application, voir la section "Exemples" du gestionnaire will finish launching (page 161).

Vous construirez un bundle dans Xcode en utilisant un de ces projets types : Application, Framework, Loadable Bundle ou Palette. Une application AppleScript Studio contient automatiquement un bundle application principal, même si vous ne suivez aucune étape pour en créer un ou pour spécifier son contenu. Ainsi qu'un fichier `Info.plist`, il contient un dossier "Scripts" (dans le dossier "Ressources") contenant les fichiers scripts compilés de l'application, tous ayant comme extension `.scpt`. Pour plus d'informations, voir la section "Exemples" ci-dessous.

Dans Mac OS X, vous pouvez examiner le contenu d'une application ayant été construite comme un bundle en faisant Control+Clic sur son icone et en choisissant "Afficher le contenu du progiciel" dans le menu contextuel.

Pour des informations supplémentaires sur le travail avec les bundles, voir la commande path for (page 112) ainsi que "Loading Resources" dans la documentation Cocoa.

Propriétés des objets de la classe Bundle

En plus des propriétés héritées de la classe item (page 60), un objet bundle possède ces propriétés :

executable path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des exécutables de l'objet bundle (par défaut, une application AppleScript Studio n'a qu'un seul exécutable)

frameworks path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des frameworks du bundle (un framework est lui-même une sorte de bundle qui regroupe le logiciel avec les ressources que celui-ci requiert, y compris son interface)

identifier

Accès : lecture uniquement
Classe : *Unicode text*
L'identificateur du bundle, vous pouvez le spécifier dans le champ "Identifier" de l'éditeur de cibles de Xcode (les détails dépendent de la version de Xcode utilisée) ; les noms des identificateurs ressemblent à `com.yourcompany.somedirectorylocation.YourAppName` ; des exemples sont visibles dans les noms des fichiers `.plist` de votre répertoire `~/Library/Preferences`

path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin du bundle ; avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser directement cette propriété dans un script,

vous deviez contourner cette limitation avec la commande call method (page 94), comme dans l'instruction suivante (laquelle obtient le chemin du bundle principal de l'application) :

```
set thePath to call method "bundlePath" of object main bundle
```

resource path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des ressources du bundle ; en fonction de l'emplacement de l'application AppleScript Studio, le chemin pourra être quelque chose comme `/Users/userName/TestApp/build/TestApp.app/Contents/Resources`

scripts path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des scripts du bundle ; en fonction de l'emplacement de l'application AppleScript Studio, le chemin pourra être quelque chose comme `/Users/userName/TestApp/build/TestApp.app/Contents/Resources/Scripts`

shared frameworks path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des frameworks partagés du bundle ; en fonction de l'emplacement de l'application AppleScript Studio, le chemin pourra être quelque chose comme `/Users/userName/TestApp/build/TestApp.app/Contents/SharedFrameworks`

shared support path

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin des supports partagés du bundle ; en fonction de l'emplacement de l'application AppleScript Studio, le chemin pourra être quelque chose comme `/Users/userName/TestApp/build/TestApp.app/Contents/SharedSupport`

Commandes supportées par les objets de la classe Bundle

Votre script peut envoyer la commande suivante à un objet bundle :

`path for` (page 112)

Events supportés par les objets de la classe Bundle

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire clicked (page 348) suivant montre comment utiliser la propriété *scripts path* pour obtenir le chemin du répertoire contenant les scripts dans le bundle principal de l'application. Il utilise la commande log (page 111) pour afficher le résultat.

```
on clicked theObject
  set thePath to scripts path of main bundle
  log thePath
end clicked
```

En fonction du nom et de l'emplacement du projet, les résultats du gestionnaire précédent ressembleront à ce qui suit :

```
2002-09-03 19:59:56.032 test project[667] "/Volumes/
Projects/test project/build/test project.app/Contents/
Resources/Scripts"
```

La section "Exemples" de la commande path for (page 112) montre comment obtenir le chemin complet et délimité par des slashs du script principal, forcément compilé, d'une application AppleScript Studio, et comment trouver et charger un script depuis le bundle principal de l'application.

Le gestionnaire clicked (page 348) suivant montre comment utiliser la commande call method (page 94) pour accéder à un bundle externe, en l'occurence ici l'application "Terminal" livrée avec Mac OS X. Il utilise la commande log (page 111) pour afficher le résultat. Une fois que la référence du bundle externe est connue, vous pouvez l'exploiter pour obtenir des informations sur ce bundle par l'intermédiaire de ses propriétés ou avec la commande path for (page 112).

Cet exemple utilise la commande call method (page 94) pour obtenir le chemin du bundle car, comme il est noté précédemment dans la section "Propriétés", la propriété *path* n'était pas supportée avant la version 1.2.1 d'AppleScript Studio. Ce gestionnaire utilise un bloc `try, on error` pour gérer le cas où la commande call method (page 94) ne serait pas capable de

retourner le bundle (par exemple, si l'application Terminal n'est pas présente dans le répertoire Utilitaires).

```
on clicked theObject
  set theBundle to call method "bundleWithPath:" of class "NSBundle"
    with parameter "/Applications/Utilities/Terminal.app"
  try
    set thePath to call method "bundlePath" of object theBundle
    log thePath
  on error
    log "Problem getting path to Terminal.app"
  end try
end clicked
```

Note : Avec AppleScript Studio version 1.2, vous pouvez écrire `of theBundle` plutôt que `of object theBundle`.

L'exemple suivant montre comment viser un bundle en dehors de l'application :

```
on clicked theObject
  set LibBundle to call method "bundleWithPath" of class "NSBundle"
    with parameter "/Users/MyUser/MyStudioLib/build/MyStudioLib.app"
  try
    tell LibBundle
      set scriptPath to path for script "MyStudioLib" extension "scpt"
    end tell
    log scriptPath
  on error
    log "Problem getting path to MyStudio.app"
  end try
end clicked
```

Entre les exemples montrés ici et l'exemple de la section "Exemples" de la commande path for (page 112), il est possible de charger un bundle externe pour y trouver et y charger des scripts. Cela signifie que vous pouvez regrouper les scripts les plus fréquemment utilisés sous une forme facilement accessible par n'importe quelle application AppleScript Studio.

Important
Lorsque vous chargez un script, vous obtenez une copie de ce script, y compris une nouvelle copie de toutes ses propriétés et variables globales. AppleScript Studio ne fournit pas par défaut de moyen satisfaisant pour partager les données entre scripts. Vous pouvez, à la rigueur, utiliser des fichiers annexes pour lire et écrire les données à transmettre, mais cette solution n'est pas pratique.

Le script suivant fournit un exemple simple sur la manière de faire.

D'abord, créez un projet AppleScript Studio dans Xcode, en utilisant un modèle AppleScript Studio. Nommez l'application "MyStudioLib" et, pour cet exemple, enregistrez-la dans votre dossier Utilisateur (`/Users/yourname/`). Dans le fichier script principal du projet, `MyStudioLib.applescript`, définissez les gestionnaires qui retourneront les scripts que vous souhaitez implémenter. Dans cet exemple, il n'y a qu'un gestionnaire, nommé `makeLibScript1`, lequel crée un script nommé `acknowledgeReceipt`. Bien qu'il n'y ait aucune instruction `return`, `makeLibScript1` retourne effectivement le script `acknowledgeReceipt`.

```
on makeLibScript1()
  script myLibScript1

  -- Handlers
    on acknowledgeReceipt()
      display dialog "The acknowledgeReceipt script greets you."
    end acknowledgeReceipt

  end script
end makeLibScript1
```

Puis, construisez le projet, lequel produit un script compilé nommé `MyStudioLib.scpt` stocké dans le bundle de l'application. Bien que cet exemple n'en fournisse qu'un seul, vous pouvez définir plusieurs gestionnaires devant retourner les scripts que vous souhaitez rendre accessible depuis votre librairie de scripts.

Finalement, vous pouvez ajouter les instructions suivantes à n'importe quel projet AppleScript Studio ayant besoin d'utiliser les scripts de votre projet MyStudioLib. Ces instructions :

– définissent des propriétés (initialisées avec la constante `missing value`) afin de rendre les scripts accessibles partout dans le fichier

- implémentent un gestionnaire `loadLibraryScripts` qui charge le fichier script depuis l'application MyStudioLib et extrait un script-objet du script
- implémentent un gestionnaire `will finish launching` qui appelle `loadLibraryScripts` lorsque l'application est lancée
- implémentent un gestionnaire `clicked` pour montrer comment appeler un script chargé ; votre application peut faire des appels identiques depuis son fichier script

```
property libraryScript : missing value
property libScript1 : missing value

on loadLibraryScripts()
  set scriptPath to missing value
  set myLibBundle to call method "bundleWithPath:" of class "NSBundle"
    with parameter "/Users/yourname/MyStudioLib/build/MyStudioLib.app"
  -- Log what we got for the bundle.
  log myLibBundle
  -- Use try, on error block to handle possible errors.
  try
    tell myLibBundle
      set scriptPath to path for script "MyStudioLib" extension "scpt"
    end tell
    set libraryScript to load script POSIX file (scriptPath)
    set libScript1 to makeLibScript1() of libraryScript
  on error
    log "Problem getting library script."
  end try
end loadLibraryScripts

on will finish launching theObject
  loadLibraryScripts()
end will finish launching

on clicked theObject
  tell libScript1 to acknowledgeReceipt()
end clicked
```

Version

Avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser directement la propriété *path* dans un script, pour contourner cette limitation, vous deviez utiliser la commande call method (page 94) comme dans la description de la propriété *path* ou dans la section "Exemples" de cette classe.

data

Pluriel : `data`
Hérite de : `item` (page 60)
Classe Cocoa : NSData

Non supportée dans la version 1.3 d'AppleScript Studio.

Version

La classe Data fut ajoutée dans la version 1.2 d'AppleScript Studio, bien qu'elle n'apporte rien dans cette version.

default entry

Pluriel : `default entries`
Hérite de : `item` (page 60)
Classe Cocoa : ASKDefaultEntry

Spécifie une inscription dans le système des valeurs utilisateurs par défaut de Mac OS X (un mécanisme permettant de stocker les valeurs par défaut sous forme de couples de clé-valeur, ou la clé est simplement un nom au format string).

Vous utiliserez cette classe dans les instructions afin d'obtenir, de régler ou de supprimer une entrée dans les valeurs par défaut d'une application, généralement utilisées pour stocker les préférences de l'utilisateur pour cette application.

Pour plus d'informations sur le système des valeurs par défaut, voir user-defaults (page 72), ainsi que User defaults dans la documentation Cocoa. Vous pouvez aussi visualiser les pages Man du système des valeurs par

défaut, en utilisant le menu “Open Man Page” du menu “Help” de Xcode (disponible à partir de Mac OS X version 10.2) pour afficher “`defaults`”, ou bien en saisissant “`man defaults`” dans une fenêtre de l’application “Terminal” (cette application est située dans le répertoire `/Applications/Utilitaires`).

Attention
Dans la version 1.2 d’AppleScript Studio, vous ne devrez pas supprimer des inscriptions depuis le système des valeurs par défaut, le faire pourrait causer le crash de votre application. Ce bug a été fixé dans la version 1.2.1.

Propriétés des objets de la classe Default Entry

En plus des propriétés héritées de la classe item (page 60), un objet default entry possède ces propriétés :

content

Accès : lecture / écriture
Classe : item (page 60)
La valeur de l’inscription ; également synonyme de *contents* ; pour plus d’informations, voir la section “Notes” de cette classe

contents

Accès : lecture / écriture
Classe : item (page 60)
La valeur de l’inscription ; également synonyme de *content* ; pour plus d’informations, voir la section “Notes” de cette classe

Events supportés par les objets de la classe Default Entry

Cette classe n’est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

La classe application (page 29) possède une propriété *user defaults*, utilisable pour manipuler les inscriptions des valeurs utilisateurs. Par exemple, vous pouvez utiliser l’instruction suivante pour créer une nouvelle inscription dans les valeurs utilisateurs. Le script d’une application AppleScript Studio n’a pas explicitement besoin de viser l’application — c’est supposé à l’intérieur du script.

```
make new default entry at end of default entries of user defaults
  with properties {name:"defaultName", contents:"Testing"}
```

Si vous essayez de faire une nouvelle inscription pour une clé déjà existante, aucune nouvelle inscription n'est créée et la valeur de la clé n'est pas modifiée. La supposition est que si la clé existe déjà, elle représente une valeur enregistrée que vous souhaitez voir préservée. Toutefois, vous pouvez modifier la valeur, si nécessaire, comme ci-dessous.

Les valeurs par défaut de votre application sont stockées dans son fichier .plist. Par exemple, si l'identificateur de votre application (lequel fut réglé dans Xcode) est "com.acme.application", le script précédent créera une nouvelle inscription "defaultName" dans le fichier avec "Testing" comme valeur (où " ∼/" indique le chemin de votre répertoire utilisateur)

∼/Library/preferences/come.acme.application.plist

Attention
Vous pouvez utiliser l'application Property List Editor, disponible dans `Application/Utilitaires/`, pour examiner les listes de propriétés des fichiers .plist. Toutefois, à cause d'un bug, certaines versions de cette application peuvent afficher incorrectement les informations des propriétés. Mais les fichiers .plist étant au format XML, vous pouvez aussi les examiner avec n'importe quel éditeur de texte.

Pour obtenir la valeur d'une inscription donnée dans les valeurs utilisateur, vous vous y référerez simplement par son nom. Par exemple, l'instruction suivante, compte tenu de l'instruction précédente`make new`, retournera la valeur "Testing" :

```
set myName to contents of default entry "defaultName" of user defaults
```

Important
La valeur d'un objet default entry est au format Unicode Text. Dans AppleScript Studio version 1.3, vous pouvez convertir une valeur en nombre, mais vous devrez d'abord la convertir en texte brut si vous voulez la convertir en valeur booléenne. Consultez la section "Discussion" ci-dessous pour plus d'informations.

Essayer d'accéder à une inscription qui n'existe pas, retournera une erreur, aussi vous devrez mettre les instructions devant accéder aux valeurs utilisateurs dans un bloc `try, on error`, comme dans l'exemple de la section "Discussion" ci-dessous.

Pour modifier une valeur, vous utiliserez la terminologie suivante :

```
set contents of default entry "defaultName" of user defaults to "Check"
```

Le gestionnaire awake from nib (page 123) suivant utilise l'instruction `tell user defaults` pour viser la propriété user-defaults (page 72) de l'objet application (page 29). Après la création du nouvel objet default entry, elle utilise une autre instruction `tell` pour logger le contenu de l'inscription, modifier son contenu, puis logger ce nouveau contenu.

```
on awake from nib theObject
  tell user defaults -- targets property of application
    make new default entry at end of default entries
      with properties {name:"test", contents:"testing"}

    tell default entry "test"
      log contents as string
      set contents to "completed"
      log contents as string
    end tell

  end tell
end awake from nib
```

Le gestionnaire précédent produira en général les logs suivants :

```
  2002-08-12 13 :46 :32.260 Test3[477] "testing"
2002-08-12 13 :46 :32.340 Test3[477] "completed"
```

Pour d'autres exemples, voir user-defaults (page 72).

Discussion

Le contenu d'un objet default entry est au format Unicode Text (comme la valeur retournée par la commande localized string (page 109)). Vous pouvez avoir besoin de convertir le format Unicode en texte brut — par exemple, si une commande d'une autre application demande du texte brut, ou pour envoyer une chaîne de caractères (comme "True" ou "False") à une valeur booléenne. Le gestionnaire suivant, extrait de l'application "SOAP Talk" distribuée avec AppleScript Studio, montre comment convertir de l'Unicode en texte brut. L'application "SOAP Talk" convertit les chaînes de caractères en texte brut car la commande `call xmlrpc` d'AppleScript n'accepte l'Unicode que depuis AppleScript 1.9.

```
on getPlainText(fromUnicodeString)
  set styledText to fromUnicodeString as string
  set styledRecord to styledText as record
  return «class ktxt» of styledRecord
end getPlainText
```

L'extrait suivant montre comment obtenir le contenu d'un objet default entry, appelle `getPlainText` pour le convertir en texte brut, et envoie le résultat à une valeur booléenne. Vous pouvez utiliser une instruction identique pour convertir une chaîne numérique en nombre. Le bloc `try...on error...` gère les possibles erreurs lors de l'obtention du contenu des inscriptions.

```
set tempString to contents of default entry "openFile" of user defaults
try
  set shouldOpen to getPlainText(tempString) as boolean
  if shouldOpen then
    -- Do whatever is needed to open.
  else
    -- Do what is needed when shouldOpen is false
  end
on error
  display dialog "Error getting should open value."
end
```

Comme alternative à la conversion obligatoire de l'Unicode en texte brut, vous pouvez comparer directement le texte, comme dans cet exemple :

```
set shouldOpen to contents of default entry "openFile" of user defaults
try
  if shouldOpen is equal to "true" then
    -- Do whatever is needed to open.
  else
    -- Do what is needed when shouldOpen is false
  end
on error
  display dialog "Error getting should open value."
end
```

Version

La classe Default Entry est apparue avec la version 1.1 d'AppleScript Studio.

Avant la version 1.2.1 d'AppleScript Studio, supprimer une valeur utilisateur pouvait provoquer le crash de votre application.

La propriété *content* est apparue avec la version 1.2 d'AppleScript Studio. Pour plus d'informations sur les différences entre *content* et *contents*, consultez la section "Version" de la classe control (page 280).

Le gestionnaire `getPlainText` fut ajouté à l'application "SOAP Talk" dans AppleScript Studio version 1.2.

event

Pluriel : `events`
Hérite de : `item` (page 60)
Classe Cocoa : NSEvent

Contient les informations sur les actions effectuées, comme un clic de souris ou l'appui sur une touche du clavier.

Chaque action de cette sorte est associée avec une fenêtre (window (page 76)), et est rapportée à l'application (application (page 29)) ayant créé la fenêtre. L'objet event contient des informations pertinentes pour chaque Event, comme l'emplacement du curseur de la souris ou quel caractère a été saisi.

Plusieurs gestionnaires d'Events, comme keyboard down (page 133), keyboard up (page 134), mouse down (page 137) et mouse up (page 141), comportent un paramètre `event` qui se réfère à l'objet event associé avec le gestionnaire. À l'intérieur de ces gestionnaires, vous pouvez utiliser ce paramètre pour accéder aux propriétés décrites ci-dessous.

Propriétés des objets de la classe Event

En plus des propriétés héritées de la classe item (page 60), un objet event possède ces propriétés :

characters

Accès : lecture seulement
Classe : *Unicode text*

Les caractères de l'Event ; plus précisément, le premier caractère saisi, comme “a”

click count

Accès : lecture seulement
Classe : *integer*
Le nombre de clic de l'Event ; “1” pour un simple clic, “2” pour un double-clic, etc

command key down

Accès : lecture seulement
Classe : *boolean*
La touche Cmd (ou Pomme) est-elle enfoncée ?

context

Accès : lecture seulement
Classe : *item* (page 60)
Le contexte d'affichage du receveur (il n'est pas recommandée d'utiliser cette propriété)

control key down

Accès : lecture seulement
Classe : *boolean*
La touche Ctrl est-elle enfoncée ?

delta x

Accès : lecture seulement
Classe : *real*
La quantité x (horizontale) de l'Event “scroll wheel”

delta y

Accès : lecture seulement
Classe : *real*
La quantité y (verticale) de l'Event “scroll wheel”

delta z

Accès : lecture seulement
Classe : *real*
La quantité z (pression) de l'Event “scroll wheel” ; utile uniquement pour les périphériques de saisie qui génèrent cette valeur, comme les stylets des tablettes graphiques

event number

Accès : lecture seulement
Classe : *integer*
Le numéro de l'Event ; il s'agit d'un compteur, peu probable que votre application en ait une réelle utilité ; pour plus d'informations, consultez la description de la méthode `event Number` de "NSEvent" dans la documentation Cocoa

event type

Accès : lecture seulement
Classe : *une des constantes* de event type (page 181)
Le type d'Event

key code

Accès : lecture seulement
Classe : *integer*
La valeur "hardware-dependent" de la touche appuyée ; il y a peu de chance que vous soyez amené à travailler avec les codes du clavier et, de plus, aucune constante n'est fournie pour les spécifier ; toutefois, vous pourriez vouloir essayer, par exemple, pour un clavier, le code pour la touche suppression est 51 et 117 pour la touche d'effacement arrière

location

Accès : lecture seulement
Classe : *point*
L'emplacement dans la fenêtre où l'Event est survenu ; l'emplacement est retourné sous la forme d'une liste de deux nombres {gauche, bas}, où par exemple, {0,0} indiquerait que l'Event a été produit dans le coin inférieur gauche de la fenêtre ; consultez la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

option key down

Accès : lecture seulement
Classe : *boolean*
La touche Alt est-elle enfoncée ?

pressure

Accès : lecture seulement
Classe : *real*
Une valeur entre 0.0 et 1.0 représentant le niveau de pression de l'ap-

pareil de saisie de l'Event (comme le stylet d'une palette graphique)

repeated

Accès : lecture seulement
Classe : *boolean*
L'Event est-il répété ?

shift key down

Accès : lecture seulement
Classe : *boolean*
La touche Maj. est-elle enfoncée ?

time stamp

Accès : lecture seulement
Classe : *real*
Le temps écoulé en secondes lorsque l'Event est survenu depuis le démarrage de l'ordinateur (par exemple, 2542.649003) ; en comparant les valeurs des propriétés *time stamp* de deux Events, vous pouvez déterminer le temps écoulé entre ces deux Events ; voir aussi la section "Exemples"

unmodified characters

Accès : lecture seulement
Classe : *Unicode text*
Les caractères non-modifiés de l'Event

window

Accès : lecture seulement
Classe : *window* (page 76)
La fenêtre associée à l'Event

Events supportés par les objets de la classe Event

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire mouse down (page 137) suivant montre comment déterminer, à l'aide du paramètre `event`, si la touche Alt était pressée lors de l'appui sur le bouton de la souris. Ainsi le gestionnaire peut utiliser cette

information pour déterminer quelles actions il doit exécuter.

```
on mouse down theObject event theEvent
  if option key down of theEvent then
    log "the option key was used"
  else
    log "the option key wasn't used"
  end if
end mouse down
```

Le gestionnaire mouse down (page 137) suivant utilise la propriété *time stamp* pour déterminer si l'utilisateur a double-cliqué. Bien sûr, vous pourriez juste connecter un gestionnaire double clicked (page 349) à l'objet si vous n'avez pas besoin de ce niveau de contrôle.

Ce gestionnaire utilise une propriété pour garder une trace de "l'heure" à laquelle est survenu le premier Event, l'initialisant avec la constante `missing value` pour indiquer qu'elle n'est pas réglée. Il est supposé, dans cet exemple, que "deux clics" fait en moins d'une seconde constitue un double-clic. Comme la valeur de la propriété *time stamp* est au format *real*, vous risquez d'obtenir une mesure du temps en fraction de secondes.

```
property lastTimeStamp : missing value

on mouse down theObject event theEvent
  if lastTimeStamp is missing value then
    set lastTimeStamp to time stamp of theEvent
  else
    if (time stamp of theEvent) - lastTimeStamp < 1 then
      display alert "You double clicked!"
    else
      set lastTimeStamp to time stamp of theEvent
    end if
  end if
end mouse down
```

font

Pluriel : `fonts`
Hérite de : `personne`
Classe Cocoa : NSFont

Non supportée dans la version 1.3 d'AppleScript Studio. Toutefois, voir la section "Exemples" pour plus d'informations sur le réglage des polices dans Interface Builder.

Exemples

Vous pouvez régler pour une police donnée, sa famille, sa taille, sa graisse et sa couleur pour l'utilisation, par exemple, dans un objet text field (page 325) ou text view (page 569), tout ceci dans le panel "Polices" d'Interface Builder, visible dans l'illustration 2.2.

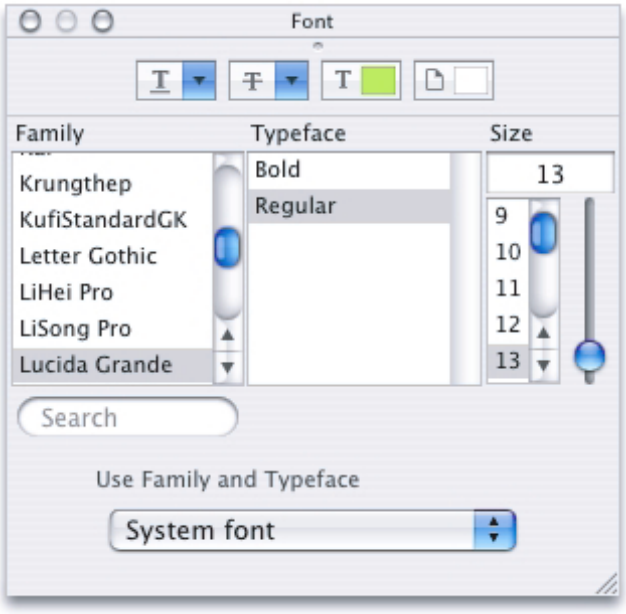

FIG. 2.2 - *Le panel "Font" dans Interface Builder*

Vous pouvez utiliser le menu déroulant "Extras..." pour prévisualiser les polices, leur taille, ouvrir le panel "Couleurs" et y choisir une couleur, ainsi que d'autres opérations. Le menu déroulant "Use Family and Typeface" vous laisse choisir parmi une sélection de polices préconfigurées.

Pour faire des modifications sur un objet spécifique, comme un text field, vous sélectionnerez cet objet, ouvrirez le panel "Polices" en naviguant dans le menu "Format" (ou en appuyant sur Cmd + T), puis vous ferez vos choix.

formatter

Pluriel : formatters
Hérite de : personne
Classe Cocoa : NSFormatter

Contrôle le format des nombres ou des dates.

Un “number formatter” contrôle le format des nombres et un “date formatter” fait la même chose mais pour les dates. Pour plus d’informations sur ces classes, voir NSNumberFormatter et NSDateFormatter dans la documentation Cocoa.

L’illustration 2.3 présente un “number formatter” que vous pouvez faire glisser depuis le panneau “Cocoa-Text” d’Interface Builder. Ces “number formatter” ou “date formatter” peuvent, par exemple, être déposés sur un champ texte dans une application AppleScript Studio pour formater le texte.

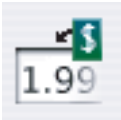

FIG. 2.3 - *Un “Number Formatter” dans Interface Builder*

L’illustration 2.4 montre le panneau “Formatter” de la fenêtre Info d’Interface Builder, dans lequel vous pourrez ajuster le format. Lorsque vous glissez un objet formatter sur un champ texte, Interface Builder affiche automatiquement le panneau “Formatter” dans la fenêtre Info. La fenêtre Info s’ouvre en appuyant sur Cmd + Maj. + I.

Events supportés par les objets de la classe Formatter

Un objet formatter supporte les gestionnaires répondant aux Events suivants. Vous pouvez suivre ces étapes pour connecter un gestionnaire à un objet formatter dans Interface Builder :

1. Mettez la fenêtre Nib de l’objet Window contenant l’objet formatter en mode Listes en cliquant sur l’icone (visible dans l’illustration 2.1 et située au-dessus de l’ascenseur vertical).
2. Utilisez le triangle pour développer l’objet window et autres jusqu’à ce que l’objet formatter soit visible.

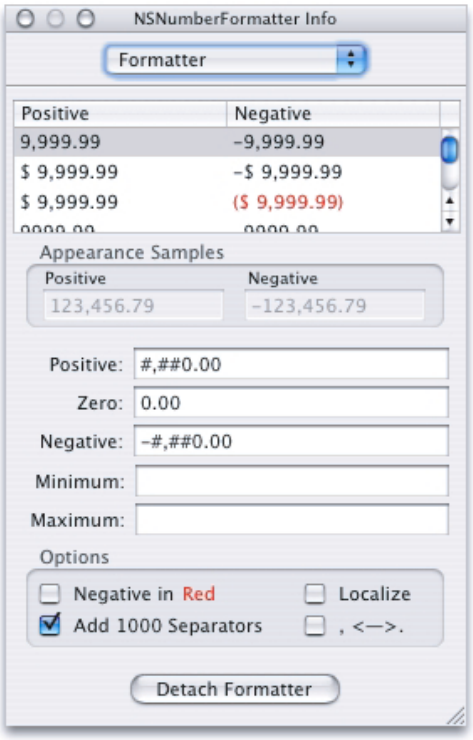

FIG. 2.4 - *La fenêtre Info d'un Number Formatter dans Interface Builder*

3. Sélectionnez l'objet formatter, enfin connectez le gestionnaire dans le panneau "AppleScript" de la fenêtre Info.

Nib

`awake from nib` (page 123)

Exemples

Dans la version 1.3 d'AppleScript Studio, les objets formatter n'ont pas de propriétés ou d'éléments scriptables. Toutefois, vous pouvez utiliser la commande call method (page 94) pour extraire des informations sur un objet formatter. Vous pouvez aussi utiliser la commande call method pour obtenir la référence d'un objet formatter d'une classe possédant une propriété *formatter*, comme control (page 280) ou cell (page 264).

La classe text field (page 325) hérite de la classe control (page 280), aussi supposons que vous ayez ajouté un objet formatter à un champ texte et l'ayez nommé "formatted", vous pourrez alors utiliser l'appel suivant pour obtenir la référence de cet objet :

```
set theFormatter to call method "formatter" of (text field "formatted" of windo
```

Supposons que le texte formaté affiché soit “$54.00” et que vous souhaitiez obtenir la chaîne de caractères exacte de ce champ texte. Le gestionnaire clicked (page 348) suivant obtient d’abord la référence de l’objet formatter, puis utilise la commande call method (page 94) pour appeler la méthode `stringForObjectValue:` de la classe Cocoa NSFormatter pour obtenir le texte formaté de cet objet formatter. Le gestionnaire utilise un bloc `try, on error` pour gérer les erreurs, et plusieurs instructions log (page 111) pour logger les diverses étapes :

```
on clicked theObject
  tell (window of theObject)
    try
      set theValue to contents of text field "formatted"
      (* get formatter, then formatted text *)
      set theFormatter to call method "formatter"
        of object (text field "formatted")
      log "Got formatter"
      set theString to call method "stringForObjectValue:"
        of object theFormatter with parameter theValue
      log ("Got string: " & theString)
    on error
      log "Error getting formatted text."
    end try
    (* Perform any operations with the formatted text. *)
  end tell
end clicked
```

Vous pouvez utiliser la commande call method (page 94) pour faire d’autres appels aux méthodes de la classe Cocoa NSFormatter.

image

Pluriel : `images`
Hérite de : `item` (page 60)
Classe Cocoa : NSImage

Représente une image.

Vous ne pouvez pas en général scripter une image — par contre vous pouvez travailler avec l'objet image view (page 285) contenant l'image.

L'objet application (page 29) possède une propriété *icon image* permettant d'accéder à l'image de l'icone, ainsi qu'aux éléments image. Les objets button (page 254) et button cell (page 261) possèdent des propriétés *image* et *alternate image*, lesquelles peuvent aussi être utilisées pour stocker des images d'icones, ou n'importe quelle autre image. Pour d'autres informations, voir Drawing and Images dans la documentation Cocoa.

L'illustration 2.5 montre l'onglet "Images" de la fenêtre MainMenu.nib d'une nouvelle application AppleScript Studio dans Interface Builder. L'image de l'icone par défaut est disponible automatiquement. Vous pouvez insérer un objet image view (page 285) dans la fenêtre d'une application en la glissant-déposant depuis le panneau "Cocoa-Controls". Vous pouvez alors glisser-déposer l'image depuis l'onglet "Images" sur l'image view. Vous pouvez aussi glisser-déposer une image sur un bouton.

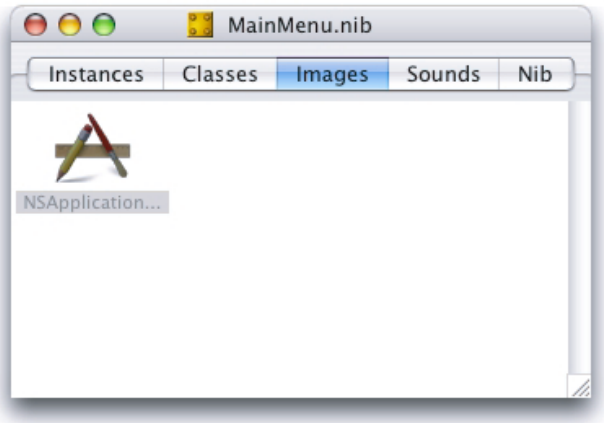

FIG. 2.5 - *L'image de l'icone de l'application dans l'onglet "Images" de la fenêtre MainMenu.nib dans Interface Builder*

Vous pouvez ajouter des images à vos applications en glissant-déposant un fichier image sur le panneau "Images" d'une fenêtre Nib dans Interface Builder, ou en glissant-déposant une liste de fichiers image sur le panneau "Users & Groups" dans le projet Xcode de l'application. Vous pouvez alors utiliser la commande load image (page 100) pour charger une image et la classe image view (page 285) pour l'afficher. Pour plus d'informations sur la façon de libérer une image (page 58), voir la section "Discussion" de la commande load image (page 100).

Events supportés par les objets de la classe Image

Bien que vous puissiez dans Interface Builder glisser-déposer une image dans une image view (page 285), vous ne pourrez pas connecter de gestionnaires d'Events à une image.

Exemples

Pour des exemples sur le travail avec un objet image, voir la classe image view (page 285).

item

Pluriel : `items`
Hérite de : `personne`
Classe Cocoa : aucune

Fournit une classe parent, avec les propriétés *name* et *ID*, à beaucoup d'autres classes. Depuis la version 1.3 d'AppleScript Studio, fournit aussi une propriété *script*.

Comme la plupart des classes héritent de la classe Item, vous pouvez attacher un script à la plupart des types d'objets ; alors, grâce à la propriété *script*, vous pouvez accéder aux propriétés, globales et gestionnaires du script. Il y a tout de même certaines limitations, par exemple, lorsque vous accédez au script d'un objet, vous obtenez une copie du script objet et non sa référence. Avec comme conséquence, si vous avez besoin de modifier la valeur d'une propriété dans un script, vous aurez besoin d'obtenir le script, de modifier la propriété, puis de régler le script en retour. Pour un exemple, voir la section "Exemples" de cette classe. Pour un exemple qui utilise la propriété *script* pour ajouter un script à un objet menu item (page 499), voir la section "Exemples" de la classe menu (page 495).

La majorité des classes d'AppleScript Studio descendent de la classe Item, soit directement, soit par l'intermédiaire de responder (page 68), view (page 229) et autres sous-classes.

La classe Item d'AppleScript Studio est différente de l'élément `item` de la classe List d'AppleScript. Vous pouvez utiliser la classe Item d'AppleScript Studio pour se référer à un objet dont vous savez qu'il hérite de cette classe ou d'une de ses sous-classes. Supposons, par exemple, que vous ayez un gestionnaire qui soit toujours transmis à un objet sous-classe de view

(page 229) ou de responder (page 68). Si le gestionnaire a uniquement besoin d'accéder aux propriétés *name* ou *ID* de l'objet transmis, il peut traiter ces objets comme des objets item. Toutefois, si vous transmettez le gestionnaire à un objet n'héritant pas de la classe Item, vous obtiendrez une erreur.

De même, vous pouvez utiliser l'élément `item` d'AppleScript pour se référer à n'importe quel élément d'une liste, bien que la liste puisse contenir différents types d'objets.

Propriétés des objets de la classe Item

Un objet item possède ces propriétés :

id

Accès : lecture uniquement
Classe : *integer*
L'ID unique de l'objet

name

Accès : lecture / écriture
Classe : *unicode text*
Le nom de l'objet ; vous fournirez un nom AppleScript à un objet dans Interface Builder, comme il est décrit dans la section "Exemples"

script

Accès : lecture / écriture
Classe : *script*
Le script associé avec l'objet ; disponible dans la version 1.3 d'AppleScript Studio ; voir la section "Exemples" de cette classe ainsi que celle de la classe menu (page 495)

Commandes supportées par les objets de la classe Item

Votre script peut envoyer la commande suivante à un objet item :

`log` (page 111)

Events supportés par les objets de la classe Item

Cette classe n'est pas accessible dans Interface Builder, par consèquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Dans les scripts AppleScript Studio, il est commode de se référer aux objets par leur nom. Par exemple :

```
set userInput to contents of text field "input" of window "main"
```

Pour fournir un nom AppleScript à un objet dans Interface Builder, vous suivrez ces étapes :

1. Avec l'objet sélectionné, ouvrez la fenêtre Info en choisissant "Show Info" dans le menu "Tools" ou en appuyant sur Cmd + Maj. + I.
2. Utilisez le menu déroulant en haut de la fenêtre Info ou (appuyez sur Cmd + 6) pour afficher le panneau "AppleScript".
3. Saisissez le nom dans le champ "Name".

Vous pouvez utiliser le script suivant dans l'Éditeur de Scripts pour obtenir les IDs de chaque view des fenêtres ouvertes d'une application "document-based application". Pour tester ce point, chaque fenêtre de document contient un certain nombre de boutons et de champs texte. Des instructions identiques fonctionneront dans le script d'une application AppleScript Studio (bien que vous n'aurez pas besoin du bloc `tell application`).

```
tell application "SimpleDocTest"
  id of every view of every window
end tell
```

Exécuter ce script avec trois fenêtres ouvertes retournera la liste suivante, contenant une liste de IDs pour chaque fenêtre :

```
{ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11},
{12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22},
{23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33} }
```

Vous pouvez remplacer `id` par `name` dans le script ci-dessus pour obtenir le nom AppleScript de chaque view (qui a un nom défini) des fenêtres ouvertes.

Pour un exemple utilisant `item` afin d'accéder aux éléments d'une liste, voir l'exemple de script de la section "Discussion" de la commande load image (page 100).

Les instructions suivantes montrent comment récupérer un script, modifier une de ses propriétés et le mettre à jour :

```
-- obtention d'une copie du script
set buttonScript to script of button 1 of window 1

-- modification de la valeur de la propriété foo
set foo of buttonScript to "new value of foo"

-- mise à jour du script avec le script modifié
set script of button 1 of window 1 to buttonScript
```

Note
Pour obtenir une propriété *script*, vous devrez explicitement utiliser `get script`.

Version

La propriété *script* fut ajoutée dans la version 1.3 d'AppleScript Studio.

movie

Pluriel : `movies`
Hérite de : `item` (page 60)
Classe Cocoa : NSMovie

Fournit une interface simple pour le chargement en mémoire des films QuickTime.

Vous ne scripterez généralement pas un objet movie lui-même. Au lieu de cela, vous travaillerez avec la classe movie view (page 296).

Les objets comme application (page 29) et movie view (page 296) possèdent des propriétés movie, mais les objets movie eux-mêmes n'ont pas de propriétés ou d'éléments scriptables.

Vous pouvez ajouter un film à votre application en glissant-déposant un fichier movie dans la liste "Files" du panneau "Users & Groups" du projet Xcode de votre application. Pour utiliser un film, vous le chargerez avec la commande load movie (page 104). Pour plus d'informations sur la libération des objets movie, voir la section "Discussion" de la commande load image (page 100).

Events supportés par les objets de la classe Movie

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Pour un exemple chargeant un objet movie dans un movie view (page 296), voir la section "Exemples" de la classe movie view (page 296).

pasteboard

Pluriel : `pasteboards`
Hérite de : `item` (page 60)
Classe Cocoa : NSPasteBoard

Fournit une interface à un "pasteboard server" supportant le transfert de données entre applications, comme les opérations copier, couper, coller ou glisser-déposer.

Les données peuvent être placées dans le pasteboard sous différentes formes. Un pasteboard est un élément de l'objet application (page 29) et est analogue au presse-papiers, excepté qu'il y a de multiples pasteboards disponibles :

- general
- font
- ruler
- find
- drag

Les pasteboards "font" et "ruler" ne sont pas utilisés à ce jour, mais pourraient l'être dans une future version. Le pasteboard "general" est le pasteboard principal. Pour obtenir le contenu du pasteboard "general", vous pouvez utiliser `contents of pasteboard "general"`. Vous utiliserez ce même format pour le pasteboard "find" (lequel est utilisé pour régler la valeur des opérations de recherche et de remplacement dans la plupart des applications). Le pasteboard "drag" est utilisé durant la gestion de l'Event glisser-déposer.

Un pasteboard donné peut contenir un certain nombre de types de format. Les types suivants sont directement supportés par AppleScript Studio :

“color”, “file”, “file names”, “font”, “html”, “image”, “pdf”, “pict image”, “postscript”, “rich text”, “rich text data”, “ruler”, “string”, “tabular text”, “url” et “vcard”.

Vous pourriez voir des types de pasteboards supplémentaires définis par le système ou par d’autres applications. Vous pouvez déterminer les types disponibles à tout moment pour un pasteboard en regardant la propriété *types*. Par exemple, si vous utilisez la phrase `types of pasteboard "general"`, vous pourriez obtenir une liste comme celle-ci {`"rich text", "string", "NeXt plain ascii pasteboard type", ...`}. Vous pourriez aussi obtenir d’autres types non définis plus haut (la plupart apparaîtront comme ceci `"CorePasteboardFlavorType 0x54455854"`).

Pour d’autres informations, voir la partie “Drag and Drop Suite” (page 475), ainsi que Cutting and Pasting, Drag and Drop et System Services dans la documentation Cocoa.

Propriétés des objets de la classe Pasteboard

En plus des propriétés qu’il hérite de la classe item (page 60), un objet pasteboard possède ces propriétés :

content

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu du pasteboard ; voir la section “Discussion”

contents

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu du pasteboard ; voir la section “Discussion”

name

Accès : lecture / écriture
Classe : *Unicode text*
Le nom du pasteboard ; une de ces valeurs : “general”, “font”, “ruler”, “find” et “drag”

preferred type

Accès : lecture / écriture
Classe : *Unicode text*
Le type de données préféré lors de l’obtention ou du réglage du contenu du pasteboard ; un des types de valeurs listés plus haut dans la des-

criptionde cette classe ; depuis la version 1.3 d'AppleScript Studio, vous pouvez utiliser la propriété *preferred type* pour régler le contenu d'un pasteboard

types

Accès : lecture seulement
Classe : *list*
Une liste des types de données supportés par le pasteboard (composée à partir des types de valeurs listés plus haut dans la description de cette classe)

Events supportés par les objets de la classe Pasteboard

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Pour obtenir les données d'un pasteboard, vous aurez besoin de régler la propriété *preferred type* du pasteboard (par défaut celle-ci sera réglée sur "string", mais cette propriété peut avoir été modifiée). Pour obtenir les données du pasteboard general au format string, vous pouvez utiliser les instructions suivantes :

```
set preferred type of pasteboard "general" to "string"
set myString to contents of pasteboard "general"
```

Pour obtenir les données d'un pasteboard, vous devrez aussi d'abord régler la propriété *preferred type* du pasteboard. Pour régler les données du pasteboard "general" au format string, vous pouvez utiliser les instructions suivantes (mais voir aussi la section "Version" de cette classe) :

```
set preferred type of pasteboard "general" to "string"
set contents of pasteboard "general" to "Testing"
```

Vous pouvez lister les pasteboards d'une application à l'aide d'un script comme celui qui suit (vous pouvez utiliser ces mêmes instructions à l'intérieur d'une application AppleScript Studio, mais vous n'aurez pas besoin du bloc `tell`) :

```
tell application "myApp"
```

```
  pasteboards
end tell
```

Ce qui suit obtiendra les types d'un pasteboard :

```
tell application "myApp"
  types of pasteboard "general"
end tell
```

Le gestionnaire awake from nib (page 123) suivant (extrait de l'application "Drag and Drop" livrée avec AppleScript Studio) utilise la commande register (page 116) pour déclarer les "drag types" auxquels un objet peut répondre :

```
on awake from nib theObject
  -- Enable support by registering the appropriate types.
  tell theObject to register drag types {"string", "rich text", "file names"}
end awake from nib
```

Depuis la version 1.3 d'AppleScript Studio, vous pouvez utiliser la propriété *preferred type* pour régler le contenu d'un pasteboard, comme avec les instructions suivantes :

```
-- In case the default wasn't already "string" :
set preferred type of pasteboard "general" to "string"
-- Now put a string on the pasteboard :
set contents of pasteboard "general" to "Testing"
```

Pour des exemples supplémentaires, voir l'application Drag and Drop.

Discussion

Vous pouvez utiliser au choix les propriétés *content* ou *contents*, sauf à l'intérieur d'un gestionnaire d'Events, `contents of theObject` retournant la référence de l'objet plutôt que son contenu. Pour obtenir le contenu d'un objet (comme le texte contenu dans un text field (page 325)) dans un gestionnaire d'Events, vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

Pour un exemple montrant la différence entre *content* et *contents*, voir la section "Version" de la classe control (page 280).

Version

Depuis la version 1.3 d'AppleScript Studio, vous pouvez utiliser la propriété *preferred type* pour régler le contenu d'un pasteboard, comme vous avez pu le lire dans la section "Exemples" de cette classe.

La classe Pasteboard et l'application "Drag and Drop" furent ajoutées dans la version 1.2 d'AppleScript Studio.

Avant la version 1.2.1 d'AppleScript Studio, vous deviez passer par une phase préparatoire avant de pouvoir régler directement le contenu d'un pasteboard avec de nouvelles données. Pour faire cela, vous invoquiez la commande call method (page 94). Les paramètres de la commande devaient être une liste de types (décrits plus haut) et le propriétaire (généralement 0 pour représenter "rien"), comme dans l'exemple suivant :

```
call method "declareTypes:owner:" of pasteboard "general"
  with parameters {{"string"}, 0}
set contents of pasteboard "general" to "Testing"
```

responder

Pluriel : `responders`
Hérite de : `item` (page 60)
Classe Cocoa : NSResponder

Fournit les bases pour le traitement des Events et les process des commandes.

Toute classe qui gère des Events doit hériter de la classe Responder, comme le font les classes application (page 29), document (page 457), window (page 76) et view (page 229).

Les applications Cocoa maintiennent une chaîne responder, laquelle lie ensemble les objets pouvant gérer les Events et les messages des actions générées par l'utilisateur. Le premier objet de la chaîne est appelé le "first responder". Les Events incluent les Events Souris et Clavier, alors que les messages des actions spécifient des actions (ou des appels aux méthodes) devant être exécutées.

Dans une application AppleScript Studio, les Events Cocoa et les messages des actions sont traduits dans les appels des gestionnaires d'Events des objets de l'application, comme keyboard down (page 133), mouse up (page 141), clicked (page 348) ou should zoom (page 156). L'opération par

défaut de la chaîne responder peut être suffisante pour la plupart des applications AppleScript Studio.

Pour certains objets, comme les objets application (page 29) et color well (page 271), les gestionnaires d'Events Clavier et Souris qui y seront connectés peuvent ne jamais être appelés, car ils sont d'abord gérés par d'autres objets avant l'objet application. Si votre application a réellement besoin de traiter avec ces gestionnaires, tenez compte de leurs connexions avec les objets présents dans l'interface utilisateur qui héritent de la classe control (page 280), comme les objets button (page 254), slider (page 315), stepper (page 320), ou text field (page 325).

Pour plus d'informations, voir Basic Event Handling dans la documentation Cocoa.

Propriétés des objets de la classe Responder

En plus des propriétés qu'il hérite de la classe item (page 60), un objet responder possède ces propriétés :

menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu de l'objet responder ; pour un objet window (page 76), cette propriété est identique à la propriété *main property* de l'objet application (page 29) ; pour les autres objets, elle sera non-définie jusqu'à ce que vous ayez ajouté un menu contextuel à cet objet

Events supportés par les objets de la classe Responder

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

La classe Responder est une classe-résumé que vous ne ciblerez pas spécifiquement dans vos scripts. Toutefois, consultez la section "Exemples" de la classe window (page 76) pour voir dans quel cas vous pourrez scripter un objet responder. Voir aussi les propriétés *first responder*, *key* et *main* de la classe window (page 76).

sound

Pluriel : sounds
Hérite de : item (page 60)
Classe Cocoa : NSSound

Représente un son pouvant être chargé et joué.

Vous ne scripterez pas généralement un son directement, mais vous pouvez utiliser la commande load sound (page 107) pour charger un objet sound et la commande play (page 338) pour le jouer. Vous pouvez jouer n'importe quels fichiers son supportés par la classe NSSound, y compris des fichiers AIFF et WAV. Pour des informations de même nature, voir Sound dans la documentation Cocoa.

L'objet application (page 29) possède des éléments sound, tandis que les objets button (page 254) et button cell (page 261) ont des propriétés *sound*.

L'illustration 2.6 montre des fichiers son dans la fenêtre Nib d'Interface Builder. Vous pouvez glisser-déposer un son depuis l'onglet "Sounds" sur un objet qui supporte les sons, comme un objet button (page 254).

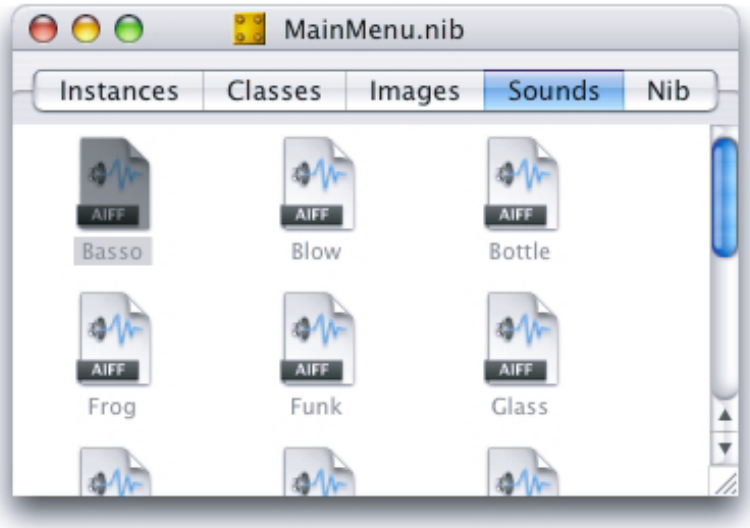

FIG. 2.6 - *Les fichiers son dans l'onglet "Sounds" d'une fenêtre Nib dans Interface Builder*

Vous pouvez ajouter des sons à votre application en les glissant-déposant dans la liste "Files" du panneau "Users & Groups" dans le projet Xcode de l'application. Actuellement pour jouer un son, vous devrez le charger avec

la commande load sound (page 107) et le jouer avec la commande play (page 338). Pour plus d'informations sur la libération des objets sound, voir la section "Discussion" de la commande load image (page 100).

Propriétés des objets de la classe Sound

En plus des propriétés héritées de la classe item (page 60), un objet sound possède ces propriétés :

playing

Accès : lecture uniquement
Classe : *boolean*
Le son est-il actuellement joué ?

Commandes supportées par les objets de la classe Sound

Votre script peut envoyer les commandes suivantes à un objet sound :

`pause` (page 337)
`play` (page 338)
`resume` (page 339)
`start` (page 340)
`stop` (page 343)

Events supportés par les objets de la classe Sound

Bien que vous puissiez glisser-déposer un son sur un objet supportant les sons dans Interface Builder, vous ne pourrez pas y connecter de gestionnaires.

Exemples

La commande load sound (page 107) fournit un exemple montrant comment une application peut charger et jouer un son. La classe slider (page 315) donne un exemple utilisant un slider laissant à l'utilisateur la possibilité de régler le volume du son.

Version

La propriété *playing* fut ajoutée dans la version 1.1 d'AppleScript Studio.

user-defaults

Pluriel : `user-defaults`
Hérite de : `item` (page 60)
Classe Cocoa : NSUserDefaults

Fournit l'accès aux valeurs utilisateurs par défaut des applications (généralement utilisées pour stocker les préférences de l'application).

Le système des valeurs utilisateurs par défaut de Mac OS X stocke les valeurs par couple "clé-valeur", où la clé est simplement une chaîne de caractères. Il y a plusieurs domaines pour les valeurs par défaut. Les valeurs par défaut du domaine global sont accessibles à n'importe quelle application. Par exemple, pendant la phase "Development" de la construction de l'application, celle-ci pourrait régler une valeur utilisateur par défaut que le débogger vérifiera pour déterminer s'il doit afficher certaines informations de déboguage. Les valeurs par défaut dans le domaine application sont généralement utilisées pour stocker les informations de préférences des applications.

Lorsqu'une application AppleScript Studio est lancée, elle "peuple" ses valeurs par défaut spécifiques avec les valeurs par défaut qu'elle peut obtenir depuis le domaine application et le domaine global, lequel contient les valeurs par défaut s'appliquant à toutes les applications d'un utilisateur. Par contre, une application ne peut pas ajouter ses propres valeurs par défaut au domaine global, mais elle peut faire des modifications qui l'emportent sur les valeurs par défaut globales chargées à l'intérieur de son propre domaine. Par exemple, l'application pourrait modifier le format par défaut des dates (voir la section "Exemples" pour plus d'informations). De telles modifications ne modifieront pas les valeurs par défaut globales pour les autres applications.

Les modifications touchant le système des valeurs par défaut sont périodiquement et automatiquement enregistrées, ainsi la prochaine fois que l'application sera lancée, les nouvelles valeurs seront présentes.

Pour accéder aux définitions des valeurs par défaut dans vos scripts AppleScript Studio, vous pouvez utiliser la propriété *user defaults* qui est associée avec chaque objet application (page 29). Notez que User-Defaults est le nom de la classe, tandis que `user defaults` spécifie un objet de cette classe.

Important
Vous ne devrez pas supprimer, dans la version 1.2 d'AppleScript Studio, des inscriptions depuis le système des valeurs par défaut. Le faire pourrait causer le crash de votre application. Ce bug a été fixé dans la version 1.2.1.

Pour plus d'informations sur le système des valeurs par défaut, voir default entry (page 45), ainsi que User Defaults dans la documentation Cocoa. Vous pouvez aussi visualiser les pages man sur le système des valeurs par défaut, en utilisant le menu "Open Man Page" dans le menu "Help" de Xcode (disponible depuis la version 10.2 de Mac OS X) pour afficher `defaults`, ou en saisissant `man defaults` dans le terminal (l'application Terminal est située dans `/Applications/Utilitaires`).

Éléments des objets de la classe User-Defaults

Un objet user-defaults peut contenir les éléments listés ci-dessous. Votre script peut spécifier la plupart des éléments par les formes-clés décrites dans "Les formes-clés standards" (page 15).

`default entry` (page 45)
spécifier par : "Les formes-clés standards" (page 15)
Les valeurs utilisateur inscrites par défaut (couples "clé-valeur")

Events supportés par les objets de la classe User-Defaults

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Vous pouvez obtenir la liste de tous les noms des valeurs utilisateur inscrites avec l'instruction suivante. La liste inclut également les valeurs que vous avez pu y ajouter, en plus de celles fournies par Mac OS, comme "AppleKeyboardUIMode" et "NSDateFormatString" :

```
set defaultsList to name of every default entry of user defaults
```

De même, vous pouvez obtenir les valeurs par défaut des inscriptions avec l'instruction suivante :

```
set defaultsContents to contents of every default entry of user defaults
```

Les instructions précédentes accèdent à la propriété *user defaults* de l'objet application (page 29). Pour plus d'informations, consulter la section "Exemples" de la classe default entry (page 45).

Pour utiliser le système des valeurs par défaut pour stocker et récupérer les préférences, votre application devra suivre ces directives :

1. Essayez de faire de nouvelles inscriptions pour toutes les préférences avant d'essayer de récupérer les réglages courants de l'utilisateur depuis le système des valeurs par défaut. Vous ferez une inscription avec une instruction telle que celle-ci :

   ```
   make new default entry at end of default entries
     of user defaults with properties
     {name : "defaultName", contents : "Testing"}
   ```

   Cette manipulation sera sans effet sur les anciennes valeurs par défaut, car si vous essayez de faire une nouvelle inscription avec une clé déjà existante, aucune inscription n'est créée et la valeur de la clé n'est pas modifiée. Voir la classe default entry (page 45) pour des informations sur la manière de modifier une inscription lorsque vous en avez besoin. L'endroit adéquat pour exécuter cette étape sera dans le gestionnaire will finish launching (page 161) connecté à votre objet application (page 29). L'objet application est représenté dans Interface Builder par l'instance File's Owner dans le panneau "Instances" de la fenêtre MainMenu.nib. Ce gestionnaire est appelé juste avant la fin du démarrage de l'application. Voir la section "Discussion" du gestionnaire awake from nib (page 123) pour plus d'informations sur l'ordre dans lequel les gestionnaires sont appelés lors du démarrage de l'application.

2. Une fois que vous avez réglé les valeurs des préférences, vous devrez essayer de lire n'importe quelle préférence utilisateur depuis le système des valeurs par défaut. Vous ferez ceci avec l'instruction suivante :

   ```
   set myName to contents of default entry "defaultName" of user defaults
   ```

   Notez que la valeur retournée est au format Unicode et que vous ne pourrez pas l'envoyer telle que à un booléen. Pour des informations sur comment obtenir une valeur booléenne, voir la section "Discussion" de la classe default entry (page 45).

3. Votre application peut à présent faire de nouvelles inscriptions ou modifier celles déjà existantes, comme les préférences de l'utilisateur.

4. Les applications Cocoa peuvent appeler la méthode `synchronize` pour spécifiquement provoquer des modifications devant être écrites dans

le système des valeurs par défaut. La commande register (page 116) d'AppleScript Studio fut à l'origine implémentée pour servir cette caractéristique, mais dans la version 1.3 d'AppleScript Studio, cette commande n'est pas fonctionnelle (bien qu'elle soit utilisée dans le support du glisser-déposer). Toutefois, le framework Cocoa appelle régulièrement la méthode `synchronize`, aussi les valeurs utilisateur par défaut sont bien enregistrées dans les applications AppleScript Studio.

Vous pouvez aussi utiliser la commande call method (page 94) pour appeler directement la méthode Cocoa `synchronize`, comme là :

```
call method "synchronize" of object user defaults
```

5. L'application devra mettre à jour son état pour refléter les modifications des préférences faites par l'utilisateur.

L'application "Archive Maker" distribuée avec AppleScript Studio depuis la version 1.1, fournit un exemple détaillé sur la manière d'utiliser le système des valeurs par défaut pour travailler avec les préférences utilisateur.

Notes

La classe User-Defaults est apparue avec la version 1.1 d'AppleScript Studio.

Depuis la version 1.2 d'AppleScript Studio, vous pouvez utiliser avec succès les listes comme type de données pour les inscriptions. Dans la version 1.1 d'AppleScript Studio, vous pouviez initialement assigner le contenu d'une inscription à une liste et la lire en retour, mais essayer d'assigner à une nouvelle liste le contenu des inscriptions par défaut ne produira pas le résultat attendu.

L'application "Archive Maker" fut d'abord distribuée avec la version 1.1 d'AppleScript Studio.

Avant la version 1.2, ce guide a listé la commande register (page 116) comme une commande supportée par la classe User-Defaults. Dorénavant, utilser la commande register (page 116) dans la version 1.2 avec un objet user-defaults ne produira rien.

window

Pluriel : `windows`
Hérite de : `responder` (page 68)
Classe Cocoa : NSWindow

Représente une fenêtre à l'écran. Un objet window gère une fenêtre à l'écran, coordonnant l'affichage et la gestion des Events de ses views. Vous pouvez créer et mettre en place des fenêtres dans Interface Builder, mais vous pouvez aussi contrôler directement plusieurs de leurs propriétés dans les scripts. L'illustration 2.7 montre une fenêtre.

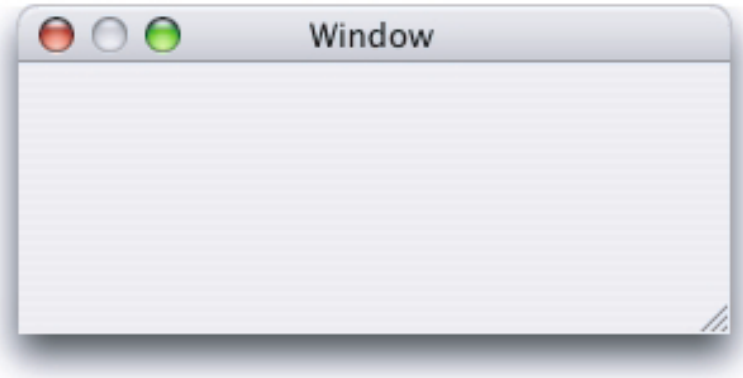

FIG. 2.7 - *Une fenêtre*

Lorsque vous créez une application AppleScript Studio grâce au modèle d'applications "AppleScript Application" de Xcode, l'application contient automatiquement par défaut une instance window, stockée dans le fichier Nib "MainMenu.nib" dans le groupe "Resources" du projet (comme dans l'illustration 2.8). Vous utiliserez ce modèle pour des applications n'ayant pas besoin de documents.

Lorsque vous créez une application AppleScript Studio grâce au modèle d'applications "AppleScript Document-based Application", l'application contient automatiquement par défaut une instance window, stockée dans le fichier .nib "Document.nib". Les applications Document-based existent afin d'autoriser l'utilisateur à créer de multiples instances de document.

N'importe quelle application est libre de définir des fichiers .nibs supplémentaires et de les utiliser pour créer une ou plusieurs instances window. Vous trouverez plusieurs objets window prédéfinis (pour des fenêtres, des panneaux et des tiroirs) dans le panneau "Cocoa-Windows" d'Interface

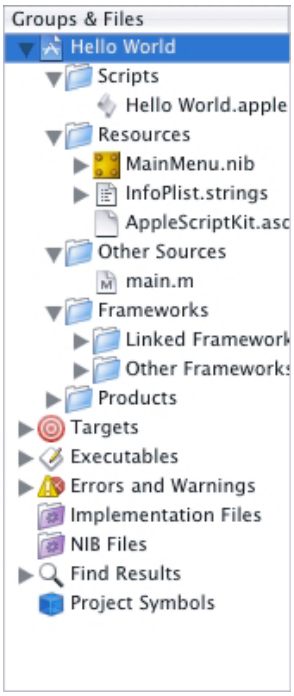

FIG. 2.8 - *La liste des fichiers du panneau "Groups & Files" dans un projet "AppleScript Application"*

Builder, comme dans l'illustration 3.3.

Dans la fenêtre Info d'Interface Builder, vous pouvez régler la plupart des attributs des fenêtres, comme les modèles de boutons qu'il contient (Miniaturize, Close et Resize), ses propriétés de taille et de redimensionnement, et si elle doit être visible au lancement. Par exemple, pour faire une fenêtre flottante (ou fenêtre utilitaire), vous utiliserez l'instance window nommée "Panel" dans l'illustration 3.3, puis ouvrirez le panneau "Attributes" de la fenêtre Info et cocherez la case "Utility window". Dans la version d'Interface Builder distribuée avec Mac OS X version 10.2, vous pouvez également régler l'attribut "Textured Window" pour spécifier à la fenêtre le look métal brossé.

Dans certains cas, comme avec un "progress panel", vous ne pourrez instancier qu'une seule fois la fenêtre, puis l'afficher et la cacher comme voulu (en utilisant soit les commandes show (page 118) et hide (page 99), soit en réglant directement la propriété *visible* de la fenêtre).

Dans d'autres cas, vous pourriez vouloir instancier de façon répétitive une nouvelle fenêtre et la libérer lorsque l'utilisateur a fini avec (ce que vous

pouvez faire en réglant la propriété *released when closed* de la fenêtre dans Interface Builder). L'application "Mail Search" distribuée avec AppleScript Studio, fournit des fichiers .nib et du code pour la création d'un "one-time status panel", ainsi qu'une fenêtre message qui est instanciée plusieurs fois. Avant la version 1.1 d'AppleScript Studio, "Mail Search" s'appelait "Watson".

Pour plus d'informations, voir Windows and Panels dans la documentation Cocoa.

Propriétés des objets de la classe Window

En plus des propriétés qu'il hérite de la classe responder (page 68), un objet window possède ces propriétés :

alpha value

Accès : lecture / écriture
Classe : *real*
La valeur alpha de la fenêtre ; une valeur de 1.0 (par défaut) indique que la fenêtre est complètement opaque, tandis que 0.0 indique que la fenêtre est complètement transparente ; l'instruction suivante règle la valeur au milieu :

```
set alpha value of window "main" to 0.5
```

associated file name

Accès : lecture / écriture
Classe : *unicode text*
Le nom du fichier associé à la fenêtre ; pour une nouvelle fenêtre non-enregistrée, cette propriété retournera une chaîne vide ; pour une fenêtre avec un fichier associé, cette propriété retournera le chemin POSIX complet (délimité par des slashs(/))

auto display

Accès : lecture / écriture
Classe : *boolean*
Affichage automatique de la fenêtre ? Avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser cette propriété dans un script

background color

Accès : lecture / écriture

Classe : *RGB color*
La couleur de fond de la fenêtre ; par défaut {65535,65535,65535} ou la couleur blanche ; non supportée avant la version 1.2 d'AppleScript Studio ; après le réglage de la propriété *background color*, la nouvelle couleur ne deviendra visible qu'une fois la fenêtre mise à jour (par script ou par interaction de l'utilisateur) ; voir la section "Exemples" pour un exemple

bounds

Accès : lecture / écriture
Classe : *bounding rectangle*
La position et la taille de la fenêtre ; les coordonnées sont représentées par une liste de quatre nombres , {gauche, bas, droite, haut} ; par exemple, {0, 0, 500, 250} indiquerait que la fenêtre a son origine dans le coin inférieur gauche de l'affichage, avec son coin supérieur droit à 500, 250.
Vous pouvez régler un emplacement ou des coordonnées avec des nombres réels (par exemple, {0.5, 0.5, 501.75, 250.1}), mais les valeurs retournées par AppleScript Studio seront toujours arrondies à leur valeur entière.
Dans ce système de coordonnées, l'origine est à gauche et en bas, et les valeurs x, y s'augmentent pour représenter respectivement la droite et le haut ; notez que c'est différent du Finder, lequel retourne les coordonnées comme ceci {gauche, haut, droite, bas}, avec l'origine dans le coin supérieur gauche, et les valeurs augmentées pour représenter le coin inférieur droit.

can hide

Accès : lecture / écriture
Classe : *boolean*
La fenêtre peut-elle être cachée ? par défaut `true` ; supplante la propriété *visible* (c'est à dire que si *can hide* vaut `false`, régler la propriété *visible* sur `false` ne cachera pas la fenêtre)

content view

Accès : lecture / écriture
Classe : *n'importe*
Le contenu de la view de la fenêtre ; la super-view de toutes les autres views de la fenêtre ; le contenu de la view est inséré automatiquement ; généralement vous ne communiquerez pas directement avec lui dans

vos scripts ; vous pourriez vouloir virer le contenu entier d'une view en modifiant son contenu — toutefois, ce n'est pas recommandé, et vous pouvez obtenir le même résultat en travaillant avec les classes tab view (page 221) et tab view item (page 227)

current text editor

Accès : lecture uniquement
Classe : *text view* (page 569)
L'éditeur de champ de la fenêtre ; voir la propriété *field editor* de la classe text (page 569) pour la description d'un éditeur

document edited

Accès : lecture / écriture
Classe : *boolean*
Le document associé à la fenêtre a-t-il été édité ? (depuis la version 1.2 d'AppleScript Studio, la classe Window a un élément document (page 457)) ; par défaut, cette propriété vaut `false` si la fenêtre n'a pas de document associé ; équivalente à la propriété *modified* de la classe document (page 457)

excluded from windows menu

Accès : lecture / écriture
Classe : *boolean*
La fenêtre doit-elle être exclue de la liste du menu "Fenêtre" ? Par défaut, cette propriété vaut `false`

first responder

Accès : lecture / écriture
Classe : *responder* (page 68)
Le premier responder de la fenêtre (le premier objet de la chaîne responder chargé de répondre aux raccourcis claviers ou autres actions) ; voir aussi la section "Exemples" de cette classe ; dans la version 1.2 d'AppleScript Studio, vous ne pouviez effectivement que régler cette propriété et l'obtenir ne retournait pas un objet bien utile ; depuis la version 1.3, la valeur retournée ressemblera à `current field editor of window 1` ; voir la propriété *field editor* de la classe text (page 569) pour la description d'un éditeur

has resize indicator

Accès : lecture / écriture
Classe : *boolean*

Faut-il que la fenêtre ait un bouton de redimensionnement ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

has shadow

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la fenêtre soit ombrée ?

hides when deactivated

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la fenêtre soit cachée lorsqu'elle est désactivée ? Si oui, passer à une autre application provoquera le masquage de la fenêtre ; par défaut, cette propriété vaut `false` ; généralement utilisée avec les fenêtres utilitaires (un type spécial de fenêtres présenté dans la description principale de cette classe) ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

key

Accès : lecture / écriture
Classe : *boolean*
La fenêtre est-elle la fenêtre clé ? La fenêtre clé est la cible courante des raccourcis claviers ; comparez les propriétés *first responder* et *main*

level

Accès : lecture / écriture
Classe : *integer*
Le niveau de la fenêtre ? Par défaut, cette propriété vaut 0 ; pour plus d'informations, voir la section "Discussion" de cette classe

main

Accès : lecture / écriture
Classe : *boolean*
La fenêtre est-elle la fenêtre principale ? La fenêtre principale est le lieu d'action de l'activité de l'utilisateur ; une fenêtre est souvent à la fois clé et principale, mais n'a pas forcément besoin de l'être ; par exemple, une fenêtre d'un document dans un traitement de texte peut être la fenêtre principale et la fenêtre clé, mais lorsque l'utilisateur ouvrira le dialogue de recherche (Cmd + F), ce dialogue deviendra la fenêtre clé ; après que l'utilisateur ait saisi le texte et réussi la recherche, la fenêtre

du document sera de nouveau la fenêtre clé et la fenêtre principale ; comparez avec la propriété *key*

maximum size

Accès : lecture / écriture
Classe : *point*
La taille maximale de la fenêtre est représentée par une liste de deux nombres (largeur maximale, hauteur maximale) ; avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser cette propriété dans un script

miniaturized

Accès : lecture / écriture
Classe : *boolean*
Non-supportée dans la version 1.3 d'AppleScript Studio ; la fenêtre est-elle miniaturisée ? (synonyme de minimized — réduite à son icone dans le dock)

minimized image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image de la fenêtre lorsqu'elle est minimisée ; les objets window peuvent ne pas avoir d'image minimisée, aussi dans ce cas vous risquez d'obtenir une valeur `missing value` si vous essayez d'obtenir cette propriété ; voir la section "Exemples" pour apprendre comment régler cette propriété

minimized title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de la fenêtre lorsqu'elle est minimisée ; ce titre apparaît lorsque vous déplacez le curseur de la souris au-dessus de l'icone de la fenêtre minimisée dans le dock ; par défaut, ce titre est identique à la propriété *title*

minimum size

Accès : lecture / écriture
Classe : *point*
La taille minimale de la fenêtre représentée par une liste de deux nombres (largeur minimale, hauteur minimale) ; avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser cette propriété dans

un script

needs display

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la fenêtre soit affichée ? Régler cette propriété sur `true` provoquera le redessinement de la fenêtre ; vous pouvez aussi utiliser la commande update (page 119) pour mettre à jour une view ; voir aussi la propriété *update display* de la classe data source (page 386). Avant la version 1.2.1 d'AppleScript Studio, vous ne pouviez pas utiliser cette propriété dans un script, mais vous pouviez utiliser la propriété *needs display* de la classe view (page 229)

opaque

Accès : lecture / écriture
Classe : *boolean*
La fenêtre est-elle opaque ? Par défaut, cette propriété vaut `true` ; Cocoa prend en compte l'opacité lorsqu'il redessine une fenêtre et ses views (pour plus d'informations voir Drawing and Images dans la documentation Cocoa), mais la plupart des applications n'auront pas besoin d'utiliser cette propriété ; pour rendre une fenêtre transparente, vous utiliserez la propriété *alpha value*

position

Accès : lecture / écriture
Classe : *point*
La position de la fenêtre ; la position est représentée par une liste de deux nombres {gauche, bas} ; par exemple, {0, 0} indiquerait le coin inférieur gauche de la fenêtre ; voir la propriété *bounds* pour plus d'informations sur le système des coordonnées

released when closed

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la fenêtre soit libérée lorsqu'elle est fermée ? Par défaut, cette propriété vaut `false` ; vous pouvez régler cette valeur dans Interface Builder ; dans certaines circonstances, vous pourriez préférer ne pas libérer la fenêtre, mais plutôt la masquer (avec la commande hide (page 99) ou en réglant sa propriété *visible* sur `false`), afin de pouvoir la réafficher de nouveau au besoin (avec la commande show (page 118) ou en réglant sa propriété *visible* sur `true`) ; une fois qu'une fenêtre

est libérée, vous devrez créer une autre instance à partir de son fichier .nib pour l'utiliser de nouveau

sheet

Accès : lecture uniquement
Classe : *boolean*
La fenêtre est-elle une feuille (sheet) ? C'est à dire, attachée à une autre fenêtre

size

Accès : lecture / écriture
Classe : *point*
La taille de la fenêtre ; la taille est représentée par une liste de deux nombres {horizontal, vertical} ; par exemple, {200, 100} indiquerait une largeur de 200 et une hauteur de 100 ; voir la propriété *bounds* pour plus d'informations sur le système des coordonnées

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de la fenêtre

visible

Accès : lecture / écriture
Classe : *boolean*
La fenêtre est-elle visible ? Par défaut, cette propriété vaut `true` pour la fenêtre principale, mais `false` pour les fenêtres supplémentaires que vous ajouterez dans Interface Builder ; vous pouvez régler cette valeur dans Interface Builder ; voir la propriété *released when closed* pour des informations de même nature ; régler la propriété *hidden* d'un objet application (page 29) sur `true` règlera la propriété *visible* de toutes ses fenêtres sur `false`, à moins que la propriété *can hide* vaut `false`, dans ce cas elle n'aura aucun effet

zoomed

Accès : lecture / écriture
Classe : *boolean*
La fenêtre est-elle agrandie ?

Éléments des objets de la classe Window

Un objet window peut contenir les éléments listés ci-dessous. Votre script peut spécifier la plupart des éléments avec n'importe laquelle des formes-clés décrites dans "Les formes-clés standards" (page 15).

`box` (page 197)
spécifier par : "Les formes-clés standards" (page 15)
Les objets box de la fenêtre

`browser` (page 363)
spécifier par : "Les formes-clés standards" (page 15)
Les objets browser de la fenêtre

`button` (page 254)
spécifier par : "Les formes-clés standards" (page 15)
Les objets button de la fenêtre

`clip view` (page 202)
spécifier par : "Les formes-clés standards" (page 15)
Les objets clip view de la fenêtre ; non-supportés par les fenêtres (dans la version 1.3 d'AppleScript Studio) ; un objet scroll view (page 213) utilise un clip view, mais sans intervention de votre application

`color well` (page 271)
spécifier par : "Les formes-clés standards" (page 15)
Les objets color well de la fenêtre

`combo box` (page 274)
spécifier par : "Les formes-clés standards" (page 15)
Les objets combo box de la fenêtre

`control` (page 280)
spécifier par : "Les formes-clés standards" (page 15)
Les objets control de la fenêtre

`document` (page 457)
spécifier par : "Les formes-clés standards" (page 15)
Les objets document de la fenêtre ; fournit l'accès au document depuis l'interface de l'application

`drawer` (page 204)
spécifier par : "Les formes-clés standards" (page 15)
Les objets drawer de la fenêtre

`image view` (page 285)
spécifier par : "Les formes-clés standards" (page 15)
Les objets image view de la fenêtre

`matrix` (page 289)
spécifier par : "Les formes-clés standards" (page 15)
Les objets matrix de la fenêtre

`movie view` (page 296)
spécifier par : "Les formes-clés standards" (page 15)
Les objets movie view de la fenêtre

`popup button` (page 301)
spécifier par : "Les formes-clés standards" (page 15)
Les objets popup button de la fenêtre

`progress indicator` (page 306)
spécifier par : "Les formes-clés standards" (page 15)
Les objets progress indicator de la fenêtre

`scroll view` (page 213)
spécifier par : "Les formes-clés standards" (page 15)
Les objets scroll view de la fenêtre

`secure text field` (page 311)
spécifier par : "Les formes-clés standards" (page 15)
Les objets secure text field de la fenêtre

`slider` (page 315)
spécifier par : "Les formes-clés standards" (page 15)
Les objets slider de la fenêtre

`split view` (page 218)
spécifier par : "Les formes-clés standards" (page 15)
Les objets split view de la fenêtre

`stepper` (page 320)
spécifier par : "Les formes-clés standards" (page 15)
Les objets stepper de la fenêtre

`tab view` (page 221)
spécifier par : "Les formes-clés standards" (page 15)
Les objets tab view de la fenêtre

`table header view` (page 402)
spécifier par : "Les formes-clés standards" (page 15)
Les objets table header view de la fenêtre

`table view` (page 403)
spécifier par : "Les formes-clés standards" (page 15)
Les objets table view de la fenêtre

`text field` (page 325)
spécifier par : "Les formes-clés standards" (page 15)
Les objets text field de la fenêtre

`text view` (page 569)
spécifier par : "Les formes-clés standards" (page 15)
Les objets text view de la fenêtre

`view` (page 229)
spécifier par : "Les formes-clés standards" (page 15)
Les objets view de la fenêtre

Commandes supportées par les objets de la classe Window

Votre script peut envoyer les commandes suivantes à un objet window :

`center` (page 98)

`close` (de la Core Suite de Cocoa, décrite dans Core Suites de la documentation Cocoa)

`hide` (page 99)

`print` (de la Core Suite de Cocoa)

`register` (page 116)

`save` (de la Core Suite de Cocoa)

`show` (page 118)

`update` (page 119)

Events supportés par les objets de la classe Window

Un objet window supporte les gestionnaires répondant aux Events suivants :

Nib

`awake from nib` (page 123)

Panel

`alert ended` (page 551)

`dialog ended` (page 552)

`panel ended` (page 553)

Fenêtre

`became key` (page 127)

`became main` (page 128)

`deminiaturized` (page 130)

`exposed` (page 131)

`miniaturized` (page 136)

`moved` (page 142)

`opened` (page 143)

`resigned key` (page 145)

`resigned main` (page 146)

`resized` (page 147)

`should close` (page 151)

`should zoom` (page 156)

`will close` (page 161)

`will miniaturize` (page 164)

`will move` (page 164)

`will open` (page 165)

`will resize` (page 167)

`will zoom` (page 169)

Exemples

Les applications pourraient avoir besoin d'exécuter des initialisations supplémentaires avant d'afficher la fenêtre principale. L'emplacement où vous pouvez faire cela est dans le gestionnaire launched (page 135), lequel est appelé lorsque l'application a fini son lancement (et après le gestionnaire awake from nib (page 123) — un autre choix possible pour l'exécution d'initialisations supplémentaires).

Vous pouvez régler la propriété *visible* d'une fenêtre sur `false` dans Interface Builder, puis la régler sur `true` dans le gestionnaire launched (page 135) (comme montré ici) pour afficher la fenêtre. Pour un exemple

plus complet, voir l'application "Assistant" distribuée avec AppleScript Studio (depuis la version 1.1). L'ordre dans lequel sont appelés les gestionnaires d'Events de l'application pendant le démarrage, y compris le gestionnaire Launched, est listé dans la description du gestionnaire awake from nib (page 123).

Ce script suppose que la fenêtre a comme nom AppleScript "main", lequel a été réglé dans le panneau "AppleScript" de la fenêtre Info d'Interface Builder.

```
on launched theObject
  -- Perform any initialization before making window visible
  -- ...
  set visible of window "main" to true
end launched
```

La plupart des classes d'interface (y compris les sous-classes de la classe control (page 280)) héritent de la classe view (page 229), laquelle possède un élément window identifiant la fenêtre contenant la view. Les gestionnaires d'Events d'AppleScript Studio ont spécifiquement un paramètre spécifiant l'objet pour lequel le gestionnaire est appelé. Si l'objet est une instance d'une classe héritant de view (page 229) (comme c'est généralement le cas), vous pouvez reprendre l'exemple suivant, présenté dans un gestionnaire clicked (page 348), pour avoir accès à la fenêtre courante :

```
on clicked theObject
  set theWindow to window of theObject
  --Use the reference to the enclosing window as needed in the handler.
end clicked
```

Pour reproduire la mise au point faite par le clavier sur un objet, comme par exemple un objet text field (page 325), vous règlerez la propriété *first responder* de sa fenêtre sur cet objet ; par exemple, vous pourriez utiliser l'instruction suivante pour désigner comme receveur de la saisie du clavier un champ texte nommé "myText" :

```
set first responder of window 1 to text field "myText" of window 1
```

Les instructions suivantes règlent la couleur de fond d'une fenêtre sur le vert, puis rendent cette nouvelle couleur visible :

```
set background color of window "main" to {0, 65535, 0}
tell window "main" to update
```

Le script suivant montre comment régler l'image minimisée d'une fenêtre. Le script choisit d'abord un fichier image, puis charge l'image et règle alors l'image minimisée de la fenêtre.

```
tell application "StudioTest"
  set imagePath to POSIX path of (choose file)
  set minImage to load image imagePath
  set minimized image of window 1 to minImage
end tell
```

Discussion

L'empilement des fenêtres dépend du rang de chacune — les fenêtres de rang supérieur sont affichées devant celles de rang inférieur ; les fenêtres de même rang peuvent être affichées l'une devant l'autre ou inversement, mais elles ne peuvent pas être affichées derrière une fenêtre de rang inférieur.

Dans la version 1.3, AppleScript Studio ne définit plus de constantes pour régler le rang des fenêtres, mais une alternative existe, le tableau 2.1 liste les valeurs courantes des constantes de rang des fenêtres Cocoa. Vous pouvez utiliser ces valeurs, mais pas les constantes, dans vos scripts. Mais n'oubliez pas que l'utilisation et le fonctionnement de ces "hard-coded values" dans vos scripts ne sont pas assurés avec les futures versions d'AppleScript Studio.

Version

Depuis la version 1.3 d'AppleScript Studio, vous pouvez accéder à la plupart des propriétés définies pour la classe Window dans la Standard Suite de Cocoa, comme *titled*.

Depuis la version 1.3 d'AppleScript Studio, la propriété *first responder* retourne un objet tel que `current field editor of window 1`. Avant elle ne retournait aucun objet utile.

La propriété *miniaturized* n'est pas supportée dans la version 1.3 d'AppleScript Studio.

L'élément `clip view` n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Le support de la propriété *current field editor* est apparu dans la version 1.2.1 d'AppleScript Studio.

Constantes	Valeur
`NSNormalWindowLevel`	0
`NSFloatingWindowLevel`	3
`NSSubmenuWindowLevel`	3
`NSTornOffMenuWindowLevel`	3
`NSModalPanelWindowLevel`	8
`NSDockWindowLevel`	20
`NSMainMenuWindowLevel`	24
`NSPopUpMenuWindowLevel`	101
`NSScreenSaverWindowLevel`	1001

TAB. 2.1 - *Les constantes de rang des fenêtres Cocoa*

Le support des propriétés suivantes fut ajoutée dans la version 1.2.1 d'AppleScript Studio (vous ne pouviez pas les utiliser dans la version 1.2) :

- *auto display*
- *maximum size*
- *minimum size*
- *needs display*

Le support de la propriété *background color* est apparu avec la version 1.2 d'AppleScript Studio.

Le support des commandes center (page 98), hide (page 99) et show (page 118) est apparu avec la version 1.2 d'AppleScript Studio.

Le support des gestionnaires will open (page 165) et will zoom (page 169) est apparu avec la version 1.2 d'AppleScript Studio.

Le support de l'attribut "Textured Window", que vous pouvez utiliser pour spécifier le look métal-brossé, est apparu avec la version d'Interface Builder distribuée avec Mac OS X version 10.2.

Chapitre 2

Commandes

Les objets basés sur les classes de la suite Application supportent les commandes suivantes. Une **commande** est un mot ou une phrase qu'un script peut utiliser pour demander une action. Pour déterminer les commandes supportées par chaque classe, voir les descriptions propres à chacune.

call method 94
center 98
hide 99
load image 100
load movie 104
load nib 106
load sound 107
localized string 109
log 111
path for 112
register 116
select 117
select all 117
show 118
size to fit 119
update 119

call method

Fournit un mécanisme pour appeler les méthodes des objets Objective-C depuis un script.

Avec la commande Call Method, vous pouvez facilement accéder à votre code Objective-C, ou utiliser les caractéristiques Cocoa non exposées dans la terminologie de scripting d'AppleScript Studio.

Voir la description de la classe document (page 457) pour des détails sur la manière d'utiliser Xcode (et un projet AppleScript Studio) pour trouver des informations sur les classes, méthodes et constantes Cocoa utilisables avec la commande Call Method.

Syntaxe

`call method`	*string*	obligatoire
`[of]`	*item*	facultatif
`[of class]`	*Unicode text*	facultatif
`[of object]`	*item*	facultatif
`[with parameter]`	*item*	facultatif
`[with parameters]`	*list*	facultatif

Paramètres

string

Le nom de la méthode à appeler

`[of]` *item* (page 60)

L'objet à envoyer à la méthode. Le paramètre `of` fut ajouté dans la version 1.2 d'AppleScript Studio, il peut être utilisé à la place du paramètre `of object` si votre application n'a pas besoin de tourner avec des versions antérieures d'AppleScript Studio.

Vous n'utiliserez jamais ensemble les paramètres `of` (ou `of object`) et `of class`. Si vous ne spécifiez aucun des deux, l'appel va à la méthode de l'objet délégué de l'application ou, si le délégué ne le supporte pas, à l'objet application (page 29) lui-même.

Plusieurs classes de Cocoa utilisent des délégués, ou des objets aide, lesquels peuvent intervenir et exécuter des opérations pour la classe utilisant ce délégué. Les délégués fournissent une manière pratique de personnaliser le comportement d'une classe sans avoir à créer une

nouvelle sous-classe. Si vous n'écrivez pas de code Cocoa, vous n'aurez probablement pas besoin de connaître les objets délégués, mais si vous êtes intéressés, vous trouverez plus d'informations dans Using Window Notifications et Delegate Methods dans la documentation Cocoa.

`[of class]` *Unicode text*

La classe à envoyer à la méthode. Vous n'utiliserez jamais ensemble `of` (ou `of object`) et `of class`.

`[of object]` *item* (page 60)

L'objet pour appeler la méthode. Si votre application a besoin de tourner avec une version plus ancienne que la 1.2 d'AppleScript Studio, utilisez `of object` au lieu de `of`.

`[with parameter]` *item* (page 60)

Spécifie un paramètre devant être transmis à la méthode appelée. Utilisez ce paramètre pour une méthode prenant qu'un seul paramètre. Vous pouvez utiliser le paramètre pour transmettre un objet ou une valeur simple comme un nombre entier. Vous pouvez aussi transmettre une liste simple, laquelle peut contenir plusieurs éléments, mais uniquement si la méthode appelée accepte un paramètre unique contenant plusieurs valeurs, comme un tableau (array) ou un dictionnaire (dictionary). "Array" et "Dictionary" sont des types Cocoa, basés sur les classes NSArray et NSDictionary.

`[with parameters]` *list*

Spécifie une liste de paramètres devant être transmise à la méthode appelée. Indiqué pour les méthodes ayant plus qu'un paramètre, bien que vous puissiez aussi l'utiliser pour une méthode avec paramètre unique. Vous spécifierez une liste avec un élément pour chaque paramètre de la méthode spécifiée. Un élément de la liste peut être également une liste, si la méthode appelée accepte un paramètre unique contenant plusieurs valeurs dans cette position.

Vous n'utiliserez jamais ensemble `with parameter` et `with parameters`. Si vous ne spécifiez aucun des deux, il est supposé que la méthode n'a pas de paramètre.

Vous devez utiliser le paramètre `with parameters` pour transmettre une valeur booléenne, même si c'est un paramètre unique. Vous transmettrez la valeur booléenne sous forme d'une liste à élément unique. Par exemple, pour régler la propriété *scrollable* d'un objet matrix (page 289), vous pourriez utiliser cette instruction :

```
call method "setScrollable:" of matrix 1 of window 1
  with parameters {true}
```

Résultat

n'importe

La valeur retournée dépend de la méthode appelée. La commande Call Method peut retourner les types Cocoa NSRect, NSPoint, NSSize et NSRange, en plus des types primitifs comme `int`, `double`, `char *`, ainsi que les pointeurs des objets Cocoa, etc. Le tableau 2.2 liste les types Cocoa généralement retournés par la commande Call Method et leur équivalent AppleScript. Il est recommandé d'utiliser un bloc `try, on error` si vous voulez par la suite exploiter le résultat de cette commande (comme dans la section "Exemples" de la commande path for (page 112)).

Type Cocoa	Équivalent AppleScript
`NSArray`	list
`NSDate`	date
`NSDictionary`	record
`NSPoint`	liste de deux nombres : {x, y}
`NSRange`	liste de deux nombres : {début offset, fin offset}
`NSRect`	liste de quatre nombres : {gauche, bas, droite, haut}
`NSSize`	liste de deux nombres : {largeur, hauteur}
`NSString`	string

TAB. 2.2 - *Les types Cocoa et leur équivalent AppleScript*

Exemples

Ce qui suit est une déclaration de méthode de la classe NSDocument de Cocoa :

```
- (BOOL)readFromFile:(NSString *) fileName ofType:(NSString *) docType
```

Cette méthode a deux paramètres, aussi pour l'appeler avec la commande Call Method, vous utiliserez l'option `with parameters`. Dans l'exemple suivant, la liste se compose de deux variables au format string (dont les valeurs ont été réglées avant l'appel) encadrées par des accolades : {`myFilenameString`, `myDocTypeString`}.

```
call method "readFromFile:ofType:" of (document 1 of window 1)
  with parameters {myFilenameString,myDocTypeString}
```

L'exemple suivant appelle la méthode `performClick:` d'un objet button (page 254), transmettant comme paramètre un autre objet button (encadré entre parenthèses car il s'agit d'une référence multi-termes).

```
call method "performClick:" of (button 1 of window 1)
  with parameter (button 2 of window 2)
```

Si votre application doit tourner avec des versions d'AppleScript Studio antérieures à la 1.2, vous devrez utiliser le paramètre `of object`. Voici comment vous feriez avec l'exemple précédent :

```
call method "performClick:" of object (button 1 of window 1)
  with parameter (button 2 of window 2)
```

L'exemple suivant appelle une des méthodes de la classe NSNumber pour obtenir en retour un objet number initialisé avec une valeur integer. Il transmet une valeur simple (le nombre 10) comme unique paramètre. Dans cet exemple, le paramètre est précis et ne requiert pas, par conséquent, de parenthèses.

```
set theResult to call method "numberWithInt:" of class "NSNumber"
  with parameter 10
```

Pour appeler la méthode de la classe NSView : `- (void) setFrame: (NSRect) frameRect`, vous utiliserez une instruction identique à la suivante (où le paramètre unique est une liste spécifiant le cadre) :

```
call method "setFrame:" of (view 1 of window 1)
  with parameter {20,20,120,120}
```

Pour plus d'exemples utilisant la commande Call Method, voir la section "Exemples" de la classe bundle (page 38) et du gestionnaire will finish launching (page 161).

Version

Le paramètre `of` est apparu avec la version 1.2 d'AppleScript Studio pour remplacer le paramètre `of object`. Les deux syntaxes sont supportées,

mais il est préférable d'utiliser `of`. Ce petit changement devrait aider à clarifier l'écriture de vos scripts. Par exemple, au lieu de `call method "title" of object (window 1)`, vous écrirez à présent `call method "title" of window 1`. Toutefois, si votre application doit tourner avec des versions plus anciennes que la 1.2, vous devrez utiliser la forme `of object`.

Depuis la version 1.2 d'AppleScript Studio, la commande Call Method supporte le type "double data". Dans les versions précédentes, "double data" était interprété comme étant une valeur integer.

La commande Call Method avait de sévères limitations dans la version 1.0 d'AppleScript Studio, y compris une mauvaise interprétation des objets spécifiés dans les paramètres `with parameter` et `with parameters`, et une incapacité à correctement retourner les objets des classes Cocoa.

center

Centre la fenêtre dans l'écran.

Pour plus d'informations sur les fenêtres, voir Windows and Panels dans la documentation Cocoa.

Syntaxe

`center` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) recevant la commande Center

Exemples

Le gestionnaire clicked (page 348) suivant, attaché à un objet button (page 254), centre l'objet window (page 76) sur lequel réside le bouton. La fenêtre est centrée en respectant le support sur lequel elle est généralement affichée.

```
on clicked theObject
  tell window of theObject to center
end clicked
```

Version

La commande Center est apparue avec la version 1.1 d'AppleScript Studio.

hide

Cache l'objet, s'il est visible.

Seuls les objets window (page 76) peuvent être cachés. Cacher une fenêtre a le même effet que de régler sa propriété *visible* sur `false`, à moins que la propriété *can hide* de la fenêtre soit réglée sur `false`, dans ce cas la commande Hide n'aura aucun effet.

Vous ne pourrez pas connecter les gestionnaires was hidden (page 159) ou will hide (page 163) à un objet window (page 76). Ces gestionnaires s'appliquent uniquement à l'objet application (page 29) et sont appelés uniquement lorsque l'utilisateur demande le masquage de la fenêtre, soit avec le menu "Masquer la fenêtre" du menu "Fenêtre", soit en appuyant sur Cmd + H.

Pour plus d'informations sur les fenêtres, voir Windows and Panels dans la documentation Cocoa.

Syntaxe

`hide` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) recevant la commande Hide

Exemples

Le gestionnaire clicked (page 348) suivant montre comment cacher une fenêtre :

```
on clicked theObject
  --Next line hides the window that contains the clicked object.
  --If you hide the current window, be sure you have a reference
  -- to it so you can make it visible again!

  tell window of theObject to hide
```

```
    --Next line would hide a window specified by name.

    tell window "second" to hide

end clicked
```

Cacher une fenêtre équivaut à régler sa propriété *visible* sur `false`. L'instruction suivante produit le même résultat que de dire à la fenêtre de se cacher (à moins que la propriété *can hide* de la fenêtre soit réglée sur `false`, dans ce cas la propriété *visible* n'aura aucun effet) :

```
set visible of window "second" to false
```

load image

Charge l'image spécifiée.

Vous chargerez une image en tant qu'objet image (page 58) et l'afficherez dans une image view (page 285). L'objet application (page 29) peut contenir des éléments image. Les classes comme button (page 254), cell (page 264), drag info (page 477), menu item (page 499) et slider (page 315) peuvent avoir des images associées.

Les types d'image supportés sont ceux de la classe Cocoa NSBitmapImageRep. Les types supportés sont JPEG, PNG, GIF, TIFF, BMP, PICT, EPS et PDF. Vous pouvez stocker des images dans votre projet AppleScript Studio avec le menu "Add Files..." du menu "Project". Vous pouvez aussi glisser des fichiers image depuis le Finder sur un des groupes (généralement le groupe "Resources") de la liste de fichiers du panneau "Groups and Files" de Xcode. Vous pouvez aussi glisser des images dans l'onglet "Images" de la fenêtre nib dans Interface Builder.

Pour plus d'informations sur le même sujet, voir Drawing and Images et Images Views dans la documentation Cocoa.

Syntaxe

`load image` *string* obligatoire

Paramètres

string

Spécifie l'image devant être chargée ; voir la section "Exemples" pour plus d'informations

Exemples

Si une image fait partie de votre projet, vous pouvez la charger en la spécifiant par son nom, sans avoir besoin de préciser l'extension. Par exemple, supposons que votre application contienne dans le progiciel un fichier nommé `starryNights.tiff`, ainsi qu'un objet image view (page 285) ayant comme nom AppleScript "artImages" et une fenêtre "artWindow", vous pouvez alors charger cette image et l'afficher dans l'image view avec les instructions suivantes :

```
set artImage to load image "starryNights"
set image of image view "artImages" of window "artWindow" to artImage
```

Vous pouvez aussi exécuter cette opération avec une seule instruction :

```
set image of image view "artImages" of window "artWindow"
  to load image "starryNights"
```

Notez que pour charger une image depuis le projet sans indiquer précisément son extension, celle-ci devra être "tiff" et non "tif".

Si l'image ne fait pas partie de votre projet, vous pouvez la charger en spécifiant le chemin POSIX de son fichier. Par exemple, si `sunFlowers.png` est stockée sur le disque dans `/User/Me/Images`, vous pouvez la charger avec l'instruction suivante :

```
set image of image view "artImages" of window "artWindow" to
load image "/User/Me/Images/sunFlowers.png"
```

Pour un exemple de suppression d'images, voir la section "Discussion" ci-dessous.

Discussion

L'objet image retourné par la commande Load Image est conservé. Dans Cocoa, tous les objets ont un "compteur de conservation". La conservation augmente le compteur ; la libération le diminue. Lorsque le compteur

atteint 0, l'objet est jeté. Un objet retourné par une des commandes de chargement a son compteur mis à 1.

Pour la plupart des objets utilisés dans une application AppleScript Studio, vous n'aurez pas besoin d'appréhender la conservation ou la libération de l'objet. Toutefois, si vous faites de multiples appels à la commande Load Image (ou load movie (page 104) ou load sound (page 107)) et que vous ne libérez pas l'image (ou le movie (page 63) ou le sound (page 70)), l'usage de la mémoire de votre application augmentera. Pour résoudre ce problème, vous pouvez explicitement supprimer l'image (ou le movie ou le sound si ce n'est pas un son système) lorsque vous avez fini avec elle. Supprimer une image (ou un movie ou un sound) la supprime de la liste d'images (et plus) gardée par l'application et la libère ; le compteur atteignant 0, l'objet est libéré. Notez que si vous supprimez une image actuellement affichée dans une image view, elle ne sera pas libérée tant que l'image view l'utilisera.

Le script suivant montre comment une application pourrait trouver toutes les images d'un certain type stockées dans cette application, puis utilise un gestionnaire `idle` pour faire défiler les images dans l'image view toutes les 2 secondes. Chaque fois qu'une image est chargée, l'image précédente est supprimée afin de libérer la mémoire utilisée.

Le gestionnaire `launched` utilise la commande call method (page 94) pour appeler une méthode du bundle (page 38) principal de l'application et obtenir la liste (stockée sous forme de propriété) de toutes les images JPEG de ce bundle. Le premier paramètre spécifie l'extension à rechercher ; le second paramètre spécifie le répertoire du bundle à scanner — transmettre une chaîne de caractères vide spécifie une recherche dans tous les répertoires. Le gestionnaire stocke le nombre d'images trouvées dans une propriété.

S'il y a plusieurs images, le gestionnaire `idle` charge une image en respectant l'ordre de la liste et le tour en question, enregistre la référence de l'ancienne image, règle la nouvelle image dans l'image view (afin de l'afficher) et libère l'ancienne image. S'il n'y a qu'une seule image, le gestionnaire `idle` ne s'enquiquine pas à essayer de la recharger. :-)))

```
property imagePaths : {}
property imageCount : 0
property imageIndex : 0

on launched theObject
  -- Get the path to all of the JPEG images in the application
  set imagePaths to call method "pathsForResourcesOfType:inDirectory:"
```

```
    of main bundle with parameters {"JPG", ""}
  try
    set imageCount to count of imagePaths
    log imageCount
  end try
end launched

on idle theObject
  -- If we have some images
  if imageCount > 0 then
    -- Only load an image if this is the first,
    -- or if we have more than one to cycle through.
    if (imageCount > 1) or (imageIndex is equal to 0) then
      -- Adjust the count
      set imageIndex to imageIndex + 1
      if imageIndex > imageCount then
        set imageIndex to 1
      end if

      -- Load the new image
      set newImage to load image (item imageIndex of imagePaths)

      -- Get a reference to the old image, if there is one
      set oldImage to image of image view "image" of window "main"

      -- Set the new image
      set image of image view "image" of window "main" to newImage

      -- Delete the old image (use try block in case no image)
      try
        delete oldImage
      end try
    end if
  end if

  -- Return 2 to call idle routine again in 2 seconds.
  return 2
end idle
```

Version

Dans la version 1.0 d'AppleScript Studio, la commande Load Image ne chargeait pas les images externes au projet de l'application. Maintenant, depuis la version 1.1, cette limitation est dépassée et Load Image charge n'importe quelle image pourvu que l'on fournisse son chemin POSIX (délimité par des slashs). Les chemins obtenus grâce à la classe bundle (page 38) sont à ce format.

Vous pouvez obtenir le chemin POSIX d'un fichier ou d'un alias en utilisant la commande `path to` et la propriété *POSIX path* fournies dans le complément de pilotage d'AppleScript (StandardAdditions.osax). Par exemple :

```
set thePath to path to desktop
  --result: alias "MacOSX:Users:BigCat:Desktop:"
set POSIXpath to POSIX path of thePath
  --result: "/Users/BigCat/Desktop/"
```

Vous pouvez examiner la terminologie des Compléments Standards d'AppleScript Studio en ouvrant le fichier `/System/Library/ScriptingAdditions/StandardAdditions.osax` avec Xcode ou l'Éditeur de Scripts, ou toute autre application capable de lire des dictionnaires de scripting.

load movie

Charge le film QuickTime spécifié.

Vous chargerez généralement un film en tant qu'objet movie (page 63) et l'afficherez dans un movie view (page 296). L'objet application (page 29) peut contenir des éléments movie.

Consultez la classe movie view (page 296) pour connaître la liste des commandes utilisables pour contrôler un film. Pour plus d'informations sur la manière de libérer un movie (page 63), voir la section "Discussion" de la commande load image (page 100).

Syntaxe

`load movie` *string* obligatoire

Paramètres

string

Spécifie le film devant être chargé

Exemples

Si un film fait partie de votre projet, vous pouvez le charger en le désignant par son nom, sans avoir besoin de préciser son extension. Par exemple, si le fichier `bdayparty4.mov` est stocké dans votre projet, que celui-ci contient une fenêtre nommée "homeMovies", elle-même contenant un objet movie view (page 296) nommé "movies", vous pouvez charger ce film et l'afficher dans le movie view avec les instructions suivantes :

```
set currentMovie to load movie "bdayparty4"
set movie of movie view "movies" of window "homeMovies" to currentMovie
```

Vous pouvez aussi exécuter cette opération avec une seule instruction :

```
set movie of movie view "movies" of window "homeMovies" to
  load movie "bdayparty4"
```

Si le film ne fait pas partie de votre projet, vous pouvez le charger en spécifiant son chemin POSIX. Par exemple, si `bdayparty4.mov` est stocké sur le disque dans `/User/Me/Movies`, vous pouvez charger ce film avec l'instruction suivante :

```
set movie of movie view "movies" of window "homeMovies" to
  load movie "/User/Me/Movies/bdayparty4.mov"
```

Version

Dans la version 1.0 d'AppleScript Studio, la commande Load Movie ne chargeait pas les films externes au projet Xcode. Maintenant, depuis la version 1.1 d'AppleScript Studio, cette limitation est dépassée et la commande Load Movie chargera n'importe quel film pourvu que l'on fournisse son chemin POSIX.

load nib

Charge le fichier .nib spécifié (ou le fichier ressource d'interface).

Depuis la version 1.1 d'AppleScript Studio, vous devez utiliser la commande Load Nib à la place de la commande load panel (page 548) pour charger un panel (comme dans la section "Exemples" ci-dessous).

Vous créerez des fichiers .nib dans Interface Builder. Pour plus d'informations sur les fichiers .nib, voir awake from nib (page 123).

Syntaxe

`load nib` *string* obligatoire

Paramètres

string

Spécifie le fichier .nib devant être chargé, sans l'extension `.nib`

Exemples

Un fichier .nib stocke la description d'un ou de plusieurs objets d'interface utilisateur, y compris la taille, l'emplacement et les connexions avec les autres objets. Charger un fichier .nib désarchive (ou crée une instance) les objets d'interface décrits dans ce type de fichier. Par exemple, l'application "Mail Search" distribuée avec AppleScript Studio définit un fichier .nib pour chaque fenêtre affichant les résultats de la recherche. Pour créer une nouvelle fenêtre Message, il fait l'appel suivant :

```
set messageWindow to makeMessageWindow()
```

Le gestionnaire `makeMessageWindow` contient le code suivant pour charger le fichier .nib. Charger le fichier .nib crée une fenêtre Message. Le gestionnaire règle alors le nom de cette fenêtre. Les résultats de ce gestionnaire sont alors affichés dans des fenêtres titrées "message1", "message2", etc.

```
on makeMessageWindow()
  load nib "Message"
  set windowCount to windowCount + 1
  set windowName to "message" & windowCount
  set name of window "message" to windowName
  return window windowName
end makeMessageWindow
```

Les instructions suivantes sont extraites du gestionnaire clicked (page 348) de l'application "Display Panel" distribuée avec AppleScript Studio. La définition de `property` se trouve en dehors du gestionnaire.

Ce script montre comment charger un panel avec la commande Load Nib. Si les réglages de la fenêtre panel n'existent pas encore, déterminés par la vérification de la propriété, le script les crée en appelant Load Nib, transmettant le nom du fichier .nib (`Settings.nib`). Le script obtient alors la référence des réglages du panel, réglés dans Interface Builder lors de la constructon du fichier .nib, en utilisant son nom AppleScript "settings".

```
property panelWindow : missing value

  -- Following is extracted from clicked handler:

  if not (exists panelWindow) then
    load nib "SettingsPanel"
    set panelWIndow to window "settings"
  end if
```

Version

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" s'appelait "Watson".

Avant la version 1.1 d'AppleScript Studio, l'application "Display Panel" utilisait la commande load panel (page 548). L'utilisation de cette commande n'est plus recommandée depuis la version 1.1 — il vaut mieux utiliser à la place la commande Load Nib, comme dans la section "Exemples" ci-dessus.

load sound

Charge le son spécifié.

Vous chargerez généralement un son en tant qu'objet sound (page 70) et le jouerez avec la commande play (page 338). L'objet application (page 29) peut contenir des éléments sound, tandis que les classes button (page 254) et button cell (page 261) possèdent des propriétés sound.

Vous pouvez jouer tous les sons supportés par la classe Cocoa NSSound, y compris les fichiers AIFF et WAV. Pour plus d'informations sur la manière de libérer un objet sound (page 70), voir la section "Discussion" de la commande load image (page 100).

Syntaxe

`load sound` *string* obligatoire

Paramètres

string

Spécifie le son devant être chargé ; la chaîne de caractères peut nommer le son dans le projet de l'application ou fournir son chemin POSIX (délimité par des slashs) ; pour plus de détails, voir la section "Exemples"

Exemples

Par défaut, un projet AppleScript Studio permet l'accès aux fichiers son système. Vous pouvez visualiser ces sons dans l'onglet "Sounds" de la fenêtre MainMenu.nib d'Interface Builder, comme dans l'illustration 2.6.

Pour pouvoir charger un son faisant partie de votre projet, celui-ci devra avoir l'extension d'un des formats supportés, comme `aif`, `aiff` ou `wav`, mais vous n'aurez pas besoin de spécifier explicitement cette extension (voir la section "Exemples" ci-dessous). Vous pouvez charger un son situé en dehors de votre projet en le spécifiant avec son chemin POSIX ; dans ce cas, vous devrez spécifier explicitement son extension.

Le gestionnaire clicked (page 348) suivant utilise le complément de pilotage `set volume` pour régler au minimum le volume sonore, puis charge et joue le son `Sosumi.aiff` situé dans `/System/Library/Sounds`. Dans cet exemple, vous n'avez pas besoin de spécifier le chemin complet du fichier son car Interface Builder fournit un accès direct aux sons système (dans l'onglet Sounds de la fenêtre MainMenu.nib).

```
on clicked theObject
  set volume 1 -- volume level goes from 0 (silent) to 7 (full volume)
  set theSound to load sound "Sosumi"
  play theSound
end clicked
```

Si vous souhaitez spécifier le chemin complet du fichier son, vous pouvez utiliser l'instruction suivante :

```
set theSound to load sound "/System/Library/Sounds/Sosumi.aiff"
```

Ce gestionnaire clicked ne libère pas le son qu'il charge. Pour plus d'informations sur la libération des objets chargés, voir la section "Discussion" de la commande load image (page 100).

Notes

La commande Load Sound est apparue avec la version 1.1 d'AppleScript Studio.

Avant la version 10.2 de Mac OS X et 1.2 d'AppleScript Studio, vous pouviez uniquement jouer un son 16-bit et non un son 8-bit, et uniquement des fichiers son ayant l'extension `.aiff`.

`localized string`

Charge la chaîne de caractères de la clé spécifiée depuis un fichier .strings (un fichier avec l'extension `.strings`).

Syntaxe

`localized string`	*string*	obligatoire
`[from table]`	*Unicode text*	facultatif
`[in bundle]`	*bundle*	facultatif

Paramètres

string
: Le nom de la clé spécifiant la chaîne de caractères à obtenir en langue locale

`[from table]` *Unicode text*
: Le nom du fichier .strings (chaque fichier .strings est représenté par un tableau) ; si vous ne spécifiez pas de fichier .strings, le fichier par défaut est le fichier `localized.strings` du projet

`[in bundle]` *bundle* (page 38)
: Le bundle (page 38) contenant le fichier .strings ; si vous ne spécifiez pas de bundle, le bundle par défaut sera le bundle de l'application

Résultats

Unicode text

La chaîne de caractères localisée de la clé spécifiée. Si la commande ne fonctionne pas, le résultat sera non-défini, aussi il est fortement conseillé d'utiliser un bloc `try, on error` si vous souhaitez utiliser par la suite ce résultat, comme dans la section "Discussion" ci-dessous

Exemples

Supposons que vous ayez deux fichiers .strings localisés dans votre projet (enregistrés au format UTF-8), l'un pour le support de la langue anglaise, l'autre pour le support de la langue française. Vous pouvez régler le format d'un fichier .strings au format UTF-8 en suivant ces étapes :

1. Sélectionnez le fichier dans la liste des fichiers du panneau "Groups and Files" dans Xcode.
2. Ouvrez la fenêtre Info, soit en appuyant sur Cmd + I, soit en choisissant "Show Info" dans le menu "Projects".
3. Dans le panneau "Text Settings" ouvert, choisissez UTF-8 dans le menu déroulant "File Encoding".

Supposons que les fichiers `Localized.strings` soient organisés comme ceci :

English.lproj/Localized.strings :

```
/* Text for the Open button */
"OPEN\_KEY" = "Open";
/* Text for the Close button */
"CLOSE\_KEY" = "Close";
```

French.lproj/Localized.strings :

```
/* Text for the Open button */
"OPEN\_KEY" = "Ouvrir";
/* Text for the Close button */
"CLOSE\_KEY" = "Fermer";
```

Vous pouvez alors utiliser la commande Localized String comme ceci :

```
get localized string "OPEN\_KEY" from table "Localized"
set theString to localized string "OPEN\_KEY" from table "Localized"
```

Cette instruction obtiendra alors la chaîne de caractères appropriée en fonction des préférences locales de l'application (le choix des langues se fait dans la fenêtre Info, rubrique Langues, de l'application dans le Finder (sélection de l'icone de l'application puis Cmd + I)).

La chaîne de caractères retournée par l'appel de la commande Localized Strings est au format Unicode text. Vous pourriez vouloir convertir cette chaîne au format texte brut — par exemple, pour l'utiliser dans une commande d'une autre application qui demande du texte brut, ou pour distribuer la chaîne retournée sous forme d'une valeur booléenne (comme `true` ou `false`). Pour un exemple montrant cela, voir la section "Discussion" de la classe default entry (page 45).

Version

La commande Localized String est apparue avec la version 1.1 d'AppleScript Studio.

log

Retourne l'objet spécifié.

La commande Log renvoie la valeur dans le panneau "Console" de l'onglet "Run" si votre application tourne dans Xcode, ou dans l'application "Console" (située dans `/Applications/Utilities`) si elle tourne dans le Finder.

La commande Log peut être extrêmement utile lors du déboggage des scripts ou peut juste servir à étudier le bon fonctionnement de votre application AppleScript Studio.

Syntaxe

`log` *reference* obligatoire

Paramètres

reference

La référence de l'objet à logger ; vous pouvez aussi fournir une chaîne de caractères à la place de la référence

Exemples

Les instructions suivantes montrent comment logger une chaîne de caractères et un objet. Le texte indiqué après les tirets (--) montre le résultat des instructions (cependant logger une chaîne de caractères produira le même résultat quelle que soit l'application).

```
on clicked theObject
log "just testing"
-- result: "just testing"
log theObject
-- result: 2002-07-23 11:42:09.274 Drawer(488) button id 2 of window id 1
-- Rest of handler not shown.
```

Vous pouvez aussi logger des variables ou des propriétés, comme dans l'exemple suivant :

```
log someCountProperty
-- result: 2002-09-17 17:04:45.596 AppName[488] 7
-- (if the value of someCountProperty is 7)
```

Pour utiliser la commande Log dans une instruction `tell` visant une application, vous pouvez utiliser cette syntaxe :

```
tell application "Finder"
  tell me to log "Entered Finder tell block."
end
```

Version

La commande Log est apparue avec la version 1.1 d'AppleScript Studio.

path for

Retourne le chemin complet de la ressource spécifiée dans le bundle visé, ou si aucun bundle n'est visé, dans le bundle principal de l'application.

Pour plus d'informations sur les bundles, y compris des exemples visant des bundles externes, voir bundle (page 38).

Syntaxe

`path for`	*reference*	obligatoire
`[column]`	*integer*	facultatif
`[directory]`	*Unicode text*	facultatif
`[extension]`	*Unicode text*	facultatif
`[localization]`	*Unicode text*	facultatif
`[resource]`	*Unicode text*	facultatif
`[script]`	*Unicode text*	facultatif

Paramètres

reference

La référence du bundle (page 38) depuis lequel doit être obtenu le chemin ; si aucun bundle n'est spécifié, le bundle principal de l'objet application (page 29) est utilisé

`[column]` *integer*

L'index de la colonne basé sur 0 du browser view ; lorsqu'est utilisé un objet browser (page 363) pour afficher un système de fichiers, vous pouvez utiliser la commande Path For pour obtenir le chemin du répertoire contenant les fichiers de cette colonne

`[directory]` *Unicode text*

Spécifie le répertoire du bundle à inspecter

`[extension]` *Unicode text*

L'extension de l'objet à rechercher

`[localization]` *Unicode text*

La version locale de la ressource à rechercher

`[resource]` *Unicode text*

Le type de ressources à rechercher

`[script]` *n'importe*

Le script à rechercher

Résultats

Unicode text

Le chemin de la ressource spécifiée. Si la commande échoue, le résultat sera non-défini, aussi il est fortement conseillé d'utiliser un bloc `try, on error` si vous souhaitez par la suite utiliser ce résultat (comme dans l'exemple ci-dessous)

Exemples

Vous pouvez utiliser le script suivant dans l'Éditeur de Scripts pour obtenir le chemin complet du script principal compilé d'une application AppleScript Studio (ici nommée "tester"). Des instructions identiques fonctionneront dans le script d'une application AppleScript Studio (bien que vous n'aurez pas besoin du bloc `tell application`). Le script spécifie la propriété *main bundle* de l'objet application (page 29) comme cible de la commande Path For.

```
tell application "tester"
  tell main bundle
  set scriptPath to path for script "tester" extension "scpt"
  end tell
end tell
```

En fonction de l'emplacement du projet, le résultat du script précédent pourra ressembler à ceci :

```
"//Volumes/Projects/tester/build/tester.app/Contents/
Resources/Scripts/tester.scpt"
```

Comme les objets bundle (page 38) et application (page 29) supportent tous les deux la commande Path For, vous pouvez simplifier le script précédent par celui qui suit. Lorsqu'aucun bundle n'est spécifié, l'application cherche automatiquement dans le bundle principal.

```
tell application "tester"
  set scriptPath to path for script "tester" extension "scpt"
end tell
```

Le gestionnaire clicked (page 348) suivant utilise la commande Path For pour obtenir le chemin du script compilé `Application.scpt` dans l'application AppleScript Studio. Comme aucun bundle n'est spécifiquement visé, la commande cherche dans le bundle principal de l'objet application (page 29). Il stocke le chemin dans une propriété et utilise un bloc `try, on error` pour gérer le cas où la commande Path For ne retournerait pas un chemin valide.

Si la commande Path For aboutit, le script utilise la commande log (page 111) pour afficher le chemin. Il utilise alors le complément de pilotage `POSIX file` pour obtenir le chemin du fichier et le complément de pilotage `load script` pour charger le script, puis assigne ce script à une propriété. À ce point, les scripts de l'application peuvent appeler les gestionnaires du script chargé.

Note
La classe `POSIX file` et la commande Load Script font partie du complément de pilotage fourni avec AppleScript, cet osax est normalement situé dans le dossier `/System/Library/ScriptingAdditions/`.

Si le script échoue, il affiche le numéro et le message d'erreur retournés. AppleScript fournit la constante `missing value` pour remplacer les informations manquantes.

```
property mainScriptPath : missing value
property theScript : missing value

on clicked theObject
  set mainScriptPath to path for script "Application" extension "scpt"
  try
    log mainScriptPath -- log the result
    set theScript to load script POSIX file (mainScriptPath)
    -- Other statements here to work with the script.
  on error errMsg number errNum
    -- Deal with any error in getting path--first log to console:
    log "Error loading script. " & "Error: " & errNum & " Msg: " & errMsg
    -- For user-related error, can display a dialog:
    display dialog "Error: " & errNum & ". " & errMsg
  end try
end clicked
```

Ce qui suit est un message possible de log généré lorsqu'une erreur survient (ici, le fichier script n'existait pas, aussi la variable `mainScriptPath` n'a pas pu être réglée) :

```
2002-10-30 16:56:44.697 on error test[512] "Error loading script. Error: -2753
Msg: The variable mainScriptPath is not defined."
```

Si vous n'êtes pas intéressé par le numéro et le message d'erreur, ou que vous n'aviez pas prévu les valeurs retournées, vous pouvez juste utiliser `on error`.

Pour d'autres exemples, voir la section "Exemples" de la classe bundle (page 38).

register

Répertorie l'objet spécifié pour recevoir les opérations de glissés (dans le sens glisser-déposer).

Pour qu'un objet puisse répondre à n'importe quel gestionnaire de Glisser-Déposer (décrits dans le chapitre "Events" (page 481)s de la partie "Drag and Drop Suite" (page 475)), vous devez répertorier les types de glissé que l'objet pourra accepter. Vous ferez cela avec la commande Register, en utilisant le paramètre `drag type` pour fournir la liste des types de "pasteboard drag" désirés. Les types de pasteboard possibles sont listés dans la classe pasteboard (page 64).

Syntaxe

`register`	*reference*	obligatoire
`[drag type]`	*list*	facultatif

Paramètres

reference
: La référence de l'objet répertorié pour recevoir les glissés

`[drag type]` *list*
: Les types de "pasteboard drag" que l'objet acceptera ; doivent être présents pour être répertorié pour les glissés ; enregistrer une liste vide videra le pasteboard et empêchera les glissés

Exemples

Le gestionnaire awake from nib (page 123) suivant répertorie deux types de glissés (`*string*` et `*file names*`) pour l'objet auquel il est connecté. Vous pourriez, par exemple, utiliser ce gestionnaire pour répertorier les types de glissés d'un objet text field (page 325).

```
on awake from nib theObject
  tell theObject to register drag types {"string", "file names"}
end awake from nib
```

Pour plus d'exemples, voir l'application "Drag and Drop" distribuée avec la version 1.2 d'AppleScript Studio.

Version

La commande Register est apparue avec la version 1.1 d'AppleScript Studio, mais elle ne produisait rien. La commande fut rendue utilisable pour le glisser-déposer avec l'ajout du paramètre `drag type` dans la version 1.2 d'AppleScript Studio.

L'application"Drag and Drop" fut distribuée à partir de la version 1.2 d'AppleScript Studio.

select

Non-supportée dans la version 1.3 d'AppleScript Studio. Sélectionne le ou les objets spécifiés.

Syntaxe

`select`	*reference*	obligatoire
`[at index]`	*integer*	facultatif
`[item]`	*item*	facultatif

Paramètres

reference
La référence de l'objet ou des objets à sélectionner

`[at index]` *integer*
L'index de l'objet à sélectionner

`[item]` *item* (page 60)
L'objet à sélectionner

select all

Non supportée dans la version 1.3 d'AppleScript Studio. Sélectionne tous les objets contenus dans l'objet spécifié.

Syntaxe

`select all`	*reference*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire est appelé

show

Montre l'objet spécifié, comme une fenêtre ou un panel, en le faisant devenir objet principal et aussi objet clé.

Montrer une fenêtre a le même effet que de régler sa propriété *visible* sur `true`.

La commande Show a deux paramètres facultatifs, `behind` et `in front of`, mais incompatibles l'un avec l'autre. Cela vous donne un certain contrôle en ce qui concerne l'ordre des fenêtres. Pour plus d'informations sur l'ordre, voir la propriété *level* et la section "Discussion" de la classe window (page 76). Pour plus d'informations sur les fenêtres, voir Windows and Panels dans la documentation Cocoa.

Syntaxe

`show`	*reference*	obligatoire
`[behind]`	*window*	facultatif
`[in front of]`	*window*	facultatif

Paramètres

reference
: La référence de l'objet à montrer

`[behind]` *window* (page 76)
: La fenêtre située juste devant celle devant être montrée (ne pas utiliser avec `in front of`)

`[in front of]` *window* (page 76)
: La fenêtre située juste derrière celle devant être montrée (ne pas utiliser avec `behind`)

Exemples

Le gestionnaire launched (page 135) suivant est extrait de l'application "XMethods Service Finder", distribuée avec la version 1.2 d'AppleScript Studio. Le gestionnaire launched est appelé vers la fin de la séquence de lan-

cement, après que les objets du fichier .nib principal aient été créés et initialisés. C'est le bon moment pour rendre visible la fenêtre principale de l'application, ce que le gestionnaire fait en appelant la commande Show. Dans cette application, la fenêtre principale a comme nom AppleScript "main".

```
on launched theObject
  show window "main"
end launched
```

Pour spécifier l'ordre des fenêtres, vous utiliserez une instruction comme celle qui suit :

```
show window "main" in front of window "settings"
```

size to fit

Non-supportée dans la version 1.3 d'AppleScript Studio. Ajuste la taille de l'objet spécifié pour l'adapter aux dimensions de son container.

Syntaxe

`size to fit` *reference* obligatoire

Paramètres

reference

La référence de l'objet à redimensionner

update

Met à jour l'affichage de l'objet window (page 76) ou view (page 229), provoquant immédiatement le redessinement de l'objet.

Syntaxe

`update` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) ou view (page 229) à mettre à jour

Exemples

Le gestionnaire launched (page 135) suivant, extrait de l'application "Browser" distribuée avec AppleScript Studio, utilise le Finder pour obtenir la liste des noms des volumes pour un objet browser (page 363), règle le délimiteur de chemin de cet objet browser, puis utilise la commande Update pour mettre à jour son affichage.

```
on launched theObject
  tell application "Finder"
    set diskNames to name of every disk
  end tell

  set path separator of browser "browser" of window "main" to ":"

  tell browser "browser" of window "main" to update
end launched
```

Chapitre 3

Events

Les objets basés sur les classes de la suite Application supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

activated 123
awake from nib 123
became key 127
became main 128
closed 129
deminiaturized 130
document nib name 130
exposed 131
idle 132
keyboard down 133
keyboard up 134
launched 135
miniaturized 136
mouse down 137
mouse dragged 138
mouse entered 139
mouse exited 139
mouse moved 140
mouse up 141

moved 142
opened 143
open untitled 143
resigned active 145
resigned key 145
resigned main 146
resized 147
right mouse down 147
right mouse dragged 149
right mouse up 150
scroll wheel 151
should close 151
should open 152
should open untitled 153
should quit 154
should quit after last window closed 155
should zoom 156
shown 157
updated 158
was hidden 159
was miniaturized 159
will become active 160
will close 161
will finish launching 161
will hide 163
will miniaturize 164
will move 164
will open 165
will quit 166
will resign active 167
will resize 167
will show 169
will zoom 169
zoomed 171

activated

Appelé après qu'un objet application (page 29) ait été activé.

Le gestionnaire peut exécuter toute opération ayant besoin de l'activation.

Le démarrage de l'application appelle dans un ordre précis certains gestionnaires d'Events, s'ils sont connectés bien sûr, dont le gestionnaire Activated, la liste complète se trouve dans la description du gestionnaire awake from nib (page 123).

Syntaxe

`activated` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) ayant été activé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Activated à un objet application (page 29), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute action requérant l'activation, comme vérifier l'état des éléments affichés dans les fenêtres de l'application.

```
on activated theObject
  (* Add script statements here to handle activation. *)
end activated
```

awake from nib

Appelé après qu'un objet ait été désarchivé de son fichier .nib, cela inclut l'instanciation de l'objet et la restauration de ses valeurs, y compris les relations avec les autres objets du fichier .nib.

L'archivage est la procédure de création d'un enregistrement détaillé d'un groupe d'objets et de valeurs liés, à partir desquels vous pouvez recréer le groupe original (par désarchivage). Pour plus d'informations sur l'archivage, voir Archiving and Serialization dans la documentation Cocoa.

Un fichier .nib est une archive d'objets et de connexions crée dans Interface Builder. Dans un gestionnaire Awake From Nib, un objet peut exécuter n'importe quelle initialisation personnalisée, une fois que tous les objets du fichier .nib aient été désarchivés et connectés, mais avant que l'interface ne soit rendue visible à l'utilisateur. Lorsqu'un objet .nib est chargé, AppleScript Studio appelle le gestionnaire Awake From Nib de chaque objet du fichier .nib lié à ce gestionnaire.

La plupart des classes qui héritent de responder (page 68), c'est à dire à peu près toutes les classes AppleScript Studio, supportent théoriquement le gestionnaire Awake From Nib. Toutefois, dans la pratique, Awake From Nib est uniquement supporté par les classes pouvant y accéder dans Interface Builder et pouvant y connecter ce gestionnaire. Pour examiner (ou connecter) dans Interface Builder les gestionnaires disponibles pour une classe, sélectionner une instance d'un objet de ce type dans le panneau "Instances" de la fenêtre Nib, puis ouvrez le panneau AppleScript de la fenêtre "Info".

L'illustration 2.9 montre le panneau AppleScript de "File's Owner" représentant l'objet application (page 29) (voir la section "Discussion" ci-dessous pour plus d'informations sur "File's Owner"). Cette instance a un seul gestionnaire connecté, le gestionnaire should quit after last window closed (page 155). Ce gestionnaire est dans le fichier `Application.applescript` du projet.

Au lancement de l'application, les gestionnaires connectés à l'objet application (page 29) sont appelés dans cet ordre (bien sûr s'ils sont connectés) :

1. will finish launching (page 161)
2. awake from nib (page 123)
3. launched (page 135)
4. will become active (page 160)
5. activated (page 123)
6. idle (page 132)

Avant qu'un quelconque gestionnaire ne soit appelé par l'objet application (page 29) (excepté le gestionnaire will finish launching (page 161) s'il est présent, lequel est toujours appelé en premier), le fichier .nib principal de l'application sera chargé, tous ses objets désarchivés, et le gestionnaire Awake From Nib appelé pour tous ses objets connectés à ce gestionnaire. Aussi un gestionnaire Awake From Nib connecté à un objet dans le fichier .nib principal de l'application, comme la fenêtre principale de l'application,

sera appelé avant tous les autres gestionnaires connectés à l'objet application (page 29) lui-même.

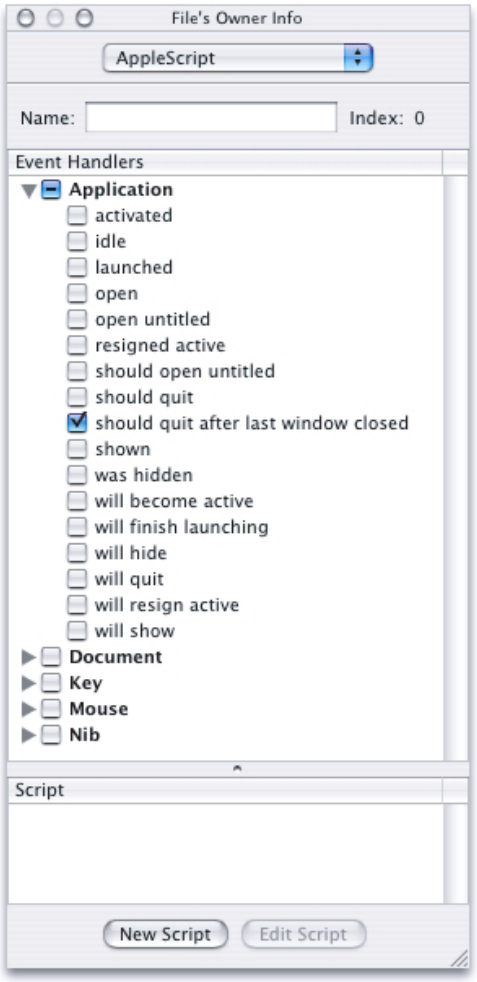

FIG. 2.9 - *La fenêtre Info d'Interface Builder, montrant les informations AppleScript de l'instance "File's Owner" d'une application*

Syntaxe

`awake from nib` *reference* obligatoire

Paramètres

reference

La référence de l'objet ayant été désarchivé

Exemples

Lorsque vous connectez un gestionnaire Awake from Nib, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge identique à celui qui suit.

```
on awake from nib theObject
  (* Perform operations here after awaking from nib. *)
end awake from nib
```

Vous pouvez connecter le gestionnaire Awake From Nib à une fenêtre et l'utiliser pour la rendre visible :

```
on awake from nib theObject
  set visible of theObject to true
end awake from nib
```

Consulter la section "Exemples" de la classe default entry (page 45) pour voir un gestionnaire Awake From Nib créant une inscription dans les valeurs utilisateur par défaut.

Discussion

Interface Builder est l'outil de création d'interface graphique d'Apple pour Mac OS X. Vous utiliserez Interface Builder pour concevoir des objets d'interface (comme des fenêtres, des contrôles, des menus, etc...), pour les redimensionner, règler leurs attributs et les connecter à d'autres objets. Les informations qui en résultent sont stockées (ou archivées) dans les ressources de l'interface utilisateur, appelées .nib, lesquelles à leur tour sont stockées dans des fichiers .nib qui deviennent partie intégrante de votre application. Un fichier .nib est un fichier Interface Builder — le "ib" de "nib" sous-entend Interface Builder.

Lorsque l'application est ouverte, elle crée une interface contenant les fenêtres, les boutons et les autres objets d'interface spécifiés dans son ou ses fichiers .nib. Une application contient aussi un fichier .nib principal qui est ouvert lorsque l'application est lancée. Elle peut aussi contenir des fichiers .nib supplémentaires et les charger au besoin, comme créer des instances de fenêtres. Pour plus d'informations sur le même sujet, voir la description de la commande load nib (page 106), ainsi que la classe document (page 457).

Lorsqu'un fichier .nib est désarchivé, il peut rétablir les connections entre les objets archivés dans ce fichier, mais pas avec les objets externes à l'archive. C'est pour cette raison qu'une application doit fournir dans Interface Builder un objet "File's Owner" à chaque fichier .nib. Pour le fichier .nib principal, montré dans l'illustration 2.1, le "File's Owner" est créé automatiquement et référence l'objet application (page 29). Dans une application "document-based AppleScript Studio", le "File's Owner" du fichier `Document.nib` est aussi créé automatiquement et référence l'objet document (page 457).

Dans Interface Builder, vous pouvez examiner la classe d'un objet "File's Owner" en sélectionnant son instance dans le panneau "Instances" de la fenêtre Nib, en ouvrant la fenêtre Info, et en utilisant le menu déroulant pour afficher le panneau "Custom Class". Par exemple, vous verrez que la classe de "File's Owner" est NSApplication pour un objet application (page 29), mais NSDocument pour un objet document (page 457). Vous pouvez modifier la classe de "File's Owner" et la régler sur une autre classe de Cocoa ou une classe personnalisée créée par vos soins, mais la plupart des applications AppleScript Studio n'auront pas besoin de cette manipulation.

Notes

Le gestionnaire Awake From Nib est apparu avec la version 1.1 d'AppleScript Studio.

Si vous travaillez avec la version d'Interface Builder distribuée avec la version 10.2 de Mac OS X, consultez la section "Information sur les versions" (page 8) pour des informations sur le réglage de la préférence "Nib File Compatibility".

`became key`

Appelé après qu'un objet window (page 76) soit devenu la fenêtre clé (ou le premier réceptacle de l'appui sur une touche du clavier).

Voir aussi became main (page 128) et resigned key (page 145), ainsi que les propriétés *key* et *main* de la classe window (page 76). Pour plus d'informations, voir Basic Event Handling dans la documentation Cocoa.

Syntaxe

`became key` *reference* obligatoire

Paramètres

reference
La référence de l'objet window (page 76) devenu l'objet clé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Became Key à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter sur la fenêtre toute opération nécessaire une fois le changement d'état (ici "clé") accompli.

```
on became key theObject
  (* Perform operations here after becoming key. *)
end became key
```

became main

Appelé lorsqu'un objet window (page 76) vient juste de devenir la fenêtre principale — c'est à dire, la fenêtre à l'avant-plan et principal lieu des actions de l'utilisateur.

La fenêtre principale n'est pas nécessairement la fenêtre clé. Voir aussi became key (page 127) et resigned main (page 146), ainsi que les propriétés *key* et *main* de la classe window (page 76). Pour plus d'informations, voir Basic Event Handling dans la documentation Cocoa.

Syntaxe

`became main` *reference* obligatoire

Paramètres

reference
La référence de l'objet window (page 76) devenu l'objet principal

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Became Main à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous

pouvez utiliser ce gestionnaire pour exécuter sur la fenêtre toute opération nécessaire une fois le changement d'état (ici "principal") accompli.

```
on became main theObject
  (* Perform operations here after becoming main. *)
end became main
```

closed

Appelé après qu'un objet drawer (page 204) (tiroir en français) soit fermé.

À partir de cet instant, le gestionnaire peut exécuter toute opération nécessaire une fois la fermeture de l'objet drawer (page 204) accomplie.

Syntaxe

`closed` *reference* obligatoire

Paramètres

reference
La référence de l'objet drawer (page 204) ayant été fermé

Exemples

L'exemple suivant est extrait de l'application "Drawer" distribuée avec AppleScript Studio.

```
on closed theObject
  set contents of text field "Date Field" of drawer "Drawer"
    of window "main" to "closed"
end closed
```

Comme *theObject* est une référence de l'objet ayant été fermé (le tiroir), l'instruction suivante est équivalente à celle écrite plus haut :

```
set contents of text field "Date Field" of theObject to "closed"
```

deminiaturized

Appelé après qu'un objet window (page 76) ait été sorti de son état "miniaturized".

Le gestionnaire peut exécuter toute opération nécessaire une fois la déréduction (un synonyme de réduction serait "mis dans le Dock") de la fenêtre accomplie.

Syntaxe

`deminiaturized` *reference* obligatoire

Paramètres

reference
La référence de l'objet window (page 76) ayant été déréduit

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Deminiaturized à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter sur la fenêtre toute opération nécessaire une fois le changement d'état (ici "déréduit") accompli, comme régler la propriété *minimized title* de la fenêtre.

```
on deminiaturized theObject
  (* Add script statements here to handle deminiaturizing. *)
end deminiaturized
```

document nib name

Retourne le nom du fichier .nib du document.

Votre application n'a pas besoin d'attacher ce gestionnaire si son "document Nib" s'appelle `document.nib`. Si vous modifiez le nom de votre "document Nib", vous devrez ajouter ce gestionnaire à l'objet application (page 29) et retourner le nom du "document Nib", sans l'extension ".nib".

Syntaxe

`document nib name`	*reference*	obligatoire
`[for document]`	*document*	obligatoire

Paramètres

reference
La référence de l'objet application (page 29)

`[for document]` *document* (page 457)
Le document dont doit être obtenu le nom de son fichier .nib

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Document Nib Name à un objet application (page 29), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Voir la description de la classe application (page 29) pour plus d'informations sur la manière d'y connecter un gestionnaire.

Vous pouvez utiliser le paramètre *theObject* pour accéder aux propriétés ou aux éléments de l'application et le paramètre `document` pour accéder aux propriétés ou aux éléments du document. Ce gestionnaire devra retourner le nom du "document Nib". Par exemple, si votre fichier "document Nib" s'appelle "MyDocument.nib", votre gestionnaire pourrait ressembler à cela :

```
on document nib name theObject for document theDocument
  (* If necessary, statements to determine name of document nib file. *)
  return "MyDocument"
end document nib name
```

Version

Le gestionnaire Document Nib Name est apparu avec la version 1.2 d'AppleScript Studio.

exposed

Non-supporté dans la version 1.3 d'AppleScript Studio. Appelé après qu'un objet window (page 76) ait été exposé pour visualisation.

Le gestionnaire peut exécuter n'importe quelle opération demandée par l'exposition de l'objet window (page 76).

Syntaxe

`exposed` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) ayant été exposé

idle

Appelé à intervalles réguliers, comme il a été défini dans l'application.

Vous utiliserez généralement un gestionnaire Idle pour exécuter de très longues ou récurantes opérations prenant place en dehors du circuit principal de l'application.

Vous connecterez un gestionaire Idle à un objet application. Consultez la description de la classe application (page 29) pour plus d'informations sur la manière de connecter un gestionnaire application. L'application, lors de sa phase de lancement, appelle certains gestionnaires, s'ils sont présents, en respectant un ordre de priorité. Le gestionnaire Idle fait partie de ces gestionnaires, la liste est disponible dans la description du gestionnaire awake from nib (page 123). Cet ordre spécifie le moment où le gestionnaire Idle est appelé. Le gestionnaire Idle retourne le nombre de secondes que l'application devra attendre avant de pouvoir à nouveau l'appeler.

Syntaxe

`idle` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) dont le gestionnaire Idle est appelé

Résultats

integer

Le nombre de secondes à attendre avant le prochain appel du gestionnaire Idle ; pour être sûr que le gestionnaire Idle sera de nouveau appelé, retournez toujours une valeur égale à 1 ou plus

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Idle à un objet application (page 29), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération requérant des appels réguliers. Votre gestionnaire devra retourner le nombre de secondes à attendre avant le prochain appel du gestionnaire Idle.

```
on idle theObject
  (* Add script statements here to perform idle operations. *)
  return 1 -- call handler again in one second
end idle
```

AppleScript fournit des constantes pour indiquer le nombre de secondes dans une minute, de minutes dans une heure, etc. Aussi pour provoquer le rappel du gestionnaire toutes les cinq minutes, vous pouvez utiliser l'instruction suivante :

```
return 5 * minutes
```

Les constantes `minutes`, `hours`, `days`, et `weeks` sont décrites dans le guide "*AppleScript Language Guide*", disponible dans l'aide de xcode et sur le site d'Apple.

Discussion

Au démarrage de l'application, un gestionnaire Idle ne pourra pas être appelé pour la première fois tant que certains autres gestionnaires, s'ils sont installés, ne l'auront pas été avant. Pour plus d'informations, voir la liste disponible dans la description du gestionnaire awake from nib (page 123).

keyboard down

Appelé lorsqu'une touche du clavier est enfoncée.

Voir la classe responder (page 68) pour plus d'informations sur la gestion par l'application des Events provoqués par la souris ou le clavier.

Syntaxe

`keyboard down`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Keyboard Down est appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "key down"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Keyboard Down, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on keyboard down theObject event theEvent
  (* Add script statements here to handle the key down event. *)
end keyboard down
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "keyboard down", comme le ou les caractères, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées. Voir la classe event (page 50) pour des exemples.

`keyboard up`

Appelé lorsqu'une touche du clavier est relâchée.

Syntaxe

`keyboard up`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Keyboard Up a été appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "key up"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Keyboard Up, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on keyboard up theObject event theEvent
  (* Add script statements here to handle the key up event. *)
end keyboard up
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "keyboard up", comme le ou les caractères, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées. Voir la classe event (page 50) pour des exemples.

Discussion

À cause de conflits dans la terminologie, ce gestionnaire ne pouvait pas être nommé `key up`.

launched

Appelé après que l'application ait été lancée.

Vous ne pouvez connecter qu'un seul gestionnaire Launched à l'objet application (page 29). Le gestionnaire peut exécuter n'importe quelle opération demandée par le lancement.

L'application, lors de sa phase de lancement, appelle certains gestionnaires, s'ils sont présents, en respectant un ordre de priorité. Le gestionnaire Launched fait partie de ces gestionnaires, la liste est disponible dans la description du gestionnaire awake from nib (page 123).

Syntaxe

`launched` *reference* obligatoire

Paramètres

reference
La référence de l'objet application (page 29) ayant été lancé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Launched à un objet application (page 29), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération rendue nécessaire après le lancement de l'application ou pouvant y prendre place. Par exemple, l'application "Drawer" distribuée avec AppleScript Studio, utilise un gestionnaire Launched pour montrer la fenêtre principale :

```
on launched theObject
  show window "main"
end launched
```

L'application "Drawer" a d'origine la propriété *visible* de sa fenêtre principale réglé sur `false`, cela a été fait lors de sa construction dans Interface Builder (dans le panneau "Attributes" de la fenêtre "Info"). Elle règle alors son interface utilisateur dans le gestionnaire awake from nib (page 123), et finalement montre la fenêtre dans le gestionnaire Launched. Consultez la section "Discussion" du gestionnaire awake from nib (page 123) pour plus d'informations sur l'ordre dans lequel les gestionnaires sont appelés lors du démarrage.

Pour d'autres exemples de gestionnaires Launched, voir la section "Exemples" de la classe application (page 29).

`miniaturized`

Appelé après qu'une fenêtre ait été réduite.

Le gestionnaire peut exécuter sur la fenêtre toute opération nécessaire une fois la réduction accomplie.

Vous devrez utiliser Miniaturized, plutôt que was miniaturized (page 159).

Syntaxe

`miniaturized` *reference* obligatoire

Paramètres

reference
La référence de l'objet window (page 76) ayant été réduit

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Miniaturized à un objet window (page 76), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois la réduction accomplie.

```
on miniaturized theObject
  (* Add script statements here to deal with miniaturizing. *)
end miniaturized
```

mouse down

Appelé lorsqu'un Event "mouse down" (enfoncement du bouton de la souris) survient et qu'il peut affecter l'objet.

Syntaxe

`mouse down`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Mouse Down a été appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "mouse down"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Down à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse down theObject event theEvent
```

```
  (* Add script statements here to handle the mouse down event. *)
end mouse down
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse down", comme l'emplacement du curseur de la souris, le nombre de clics, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

mouse dragged

Appelé lorsqu'un Event "mouse dragged" (un glisser avec la souris) survient et qu'il peut affecter l'objet.

Syntaxe

`mouse dragged`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
: La référence de l'objet dont le gestionnaire Mouse Dragged a été appelé

`[event]` *event* (page 50)
: Les informations d'Events de l'Event "mouse dragged"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Dragged à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse dragged theObject event theEvent
  (* Add script statements here to handle the mouse dragged event. *)
end mouse dragged
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse dragged", comme l'emplacement du curseur de la souris, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

mouse entered

Appelé lorsqu'un Event "mouse entered" (entrée du curseur de la souris) survient et qu'il peut affecter l'objet.

C'est à dire que le curseur est entré dans les limites de l'objet connecté au gestionnaire. La plupart des classes qui héritent des classes control (page 280) et view (page 229) supportent le gestionnaire Mouse Entered.

Syntaxe

`mouse entered`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Mouse Entered est appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "mouse entered"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Entered à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse entered theObject event theEvent
  (* Add script statements here to handle the mouse entered event. *)
end mouse entered
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse entered", comme l'emplacement du curseur de la souris, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

mouse exited

Appelé lorsqu'un Event "mouse exited" (sortie du curseur de la souris) survient et qu'il peut affecter l'objet.

C'est à dire que le curseur est sorti des limites de l'objet connecté au gestionnaire. La plupart des classes qui héritent des classes control (page 280) et view (page 229) supportent le gestionnaire Mouse Exited.

Syntaxe

`mouse exited`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference

La référence de l'objet dont le gestionnaire Mouse Exited est appelé

`[event]` *event* (page 50)

Les informations d'Events de l'Event "mouse exited"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Exited à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse exited theObject event theEvent
  (* Add script statements here to handle the mouse exited event. *)
end mouse exited
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse exited", comme l'emplacement du curseur, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

mouse moved

Appelé lorsque la souris est déplacée à l'intérieur des limites de l'objet.

C'est à dire que le curseur de la souris est déplacée à l'intérieur des limites de l'objet connecté au gestionnaire. La plupart des classes qui héritent des classes control (page 280) et view (page 229) supportent le gestionnaire Mouse Moved.

Syntaxe

`mouse moved`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Mouse Moved est appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "mouse moved"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Moved à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse moved theObject event theEvent
  (* Add script statements here to handle the mouse moved event. *)
end mouse moved
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse moved", comme l'emplacement du curseur, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

mouse up

Appelé lorsqu'un Event "mouse up" (relâchement du bouton de la souris) survient et qu'il peut affecter l'objet.

Syntaxe

`mouse up`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Mouse Up est appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "mouse up"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Mouse Up à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on mouse up theObject event theEvent
  (* Add script statements here to handle the mouse up event. *)
end mouse up
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "mouse up", comme l'emplacement du curseur, le nombre de clics, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

`moved`

Appelé après que l'objet ait été déplacé.

Syntaxe

`moved` *reference* obligatoire

Paramètres

reference
La référence de l'objet ayant été déplacé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Moved à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre *theObject* référence l'objet ayant été déplacé, généralement un objet view (page 229) ou une sous-classe de view. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois le déplacement réalisé. Pour déterminer de combien l'objet a été déplacé, vous devrez stocker son ancien emplacement et le comparer avec le nouvel emplacement.

```
on moved theObject
  (* Add script statements here to handle operations after a move. *)
end moved
```

opened

Appelé après qu'un objet supportant ce gestionnaire (comme une fenêtre, un panel ou un document) soit ouvert. À cet instant, le gestionnaire peut exécuter toute opération pouvant y prendre place.

Syntaxe

`opened` *reference* obligatoire

Paramètres

reference
La référence de l'objet ayant été ouvert

Exemples

L'exemple suivant de gestionnaire Opened est extrait de l'application "Drawer" distribuée avec AppleScript Studio.

```
on opened theObject
  set contents of text field "Date Field" of drawer "Drawer"
    of window "main" to "opened"
end opened
```

Comme *theObject* est la référence de l'objet ayant été ouvert (Drawer ou tiroir en français), l'instruction suivante est équivalente à celle écrite plus haut :

```
set contents of text field "Date Field" of theObject to "opened"
```

open untitled

Appelé lorsque l'application est sur le point d'ouvrir un document "sans titre" (untitled).

Ce gestionnaire est uniquement appelé pour une application "document-based application" et seulement pour le premier document lorsque l'application est lancée. Ce gestionnaire est appelé après le gestionnaire should open untitled (page 153) et peut préparer la fenêtre "sans titre" devant être ouverte.

Syntaxe

`open untitled` *reference* obligatoire

Paramètres

reference

La référence de l'application (page 29) ouvrant un objet document (page 457) "sans titre"

Exemples

Vous connecterez dans Interface Builder un gestionnaire Open Untitled en sélectionnant l'instance "File's Owner" dans la fenêtre MainMenu.nib de l'application, puis en sélectionnant Open Untitled parmi les gestionnaires application de la fenêtre Info. Lorsque vous sélectionnerez le script devant accueillir ce gestionnaire, AppleScript Studio insérera automatiquement un gestionnaire vierge identique à celui ci-dessous.

L'instance "File's Owner" de la fenêtre Nib principale représente NSApp, une constante globale qui référence l'objet NSApplication de l'application. Dans un fichier .nib document, l'instance "File's Owner" représente généralement le document.

Le paramètre *theObject* du gestionnaire Open Untitled référence l'objet application (page 29) pour lequel un document sans titre est sur le point d'être ouvert. Vous pouvez utiliser ce paramètre pour accéder aux propriétés ou aux éléments de l'application afin de préparer l'ouverture du document.

```
on open untitled theObject
  (* Perform operations here before opening an untitled document. *)
end open untitled
```

resigned active

Appelé après que l'objet application (page 29) ait abandonné son état actif.

Il n'y a pas de gestionnaire qui permette à l'application de refuser l'abandon de son état actif. Voir aussi will resign active (page 167).

Syntaxe

`resigned active` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) ayant abandonné son état actif

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Resigned Active à un objet application (page 29), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois que l'objet application (page 29) a abandonné son état actif.

```
on resigned active theObject
  (* Perform operations here after resigning active state. *)
end resigned active
```

resigned key

Appelé après qu'une fenêtre ait abandonné son état clé (par exemple, comme premier réceptacle des touches du clavier).

Voir aussi became key (page 127) et will resign active (page 167).

Syntaxe

`resigned key` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) ayant abandonné son état d'objet clé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Resigned Key à un objet window (page 76), AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois que la fenêtre a abandonné son état d'objet clé.

```
on resigned key theObject
  (* Perform operations here after resigning key state. *)
end resigned key
```

resigned main

Appelé après qu'un objet window (page 76) ait abandonné son état principal (comme la fenêtre en avant-plan et principal centre d'action des opérations de l'utilisateur).

Une fenêtre peut être principale sans être clé (premier réceptacle des touches du clavier). Voir aussi became main (page 128), became key (page 127) et resigned key (page 145).

Syntaxe

`resigned main` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) ayant abandonné son état d'objet principal

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Resigned Main à un objet window (page 76), AppleScript Studio ajoute automati-

quement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois que la fenêtre a abandonné son état d'objet principal.

```
on resigned main theObject
  (* Perform operations here after resigning main. *)
end resigned main
```

resized

Appelé après qu'un objet ait été redimensionné.

Syntaxe

`resized` *reference* obligatoire

Paramètres

reference
La référence de l'objet ayant été redimensionné

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Resized à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois le redimensionnement accompli. Pour mesurer la modification, vous devrez stocker l'ancienne taille de l'objet et la comparer avec sa nouvelle taille. Pour faire cela, vous pouvez sauvegarder la taille dans un gestionnaire will resize (page 167), lequel sera appelé avant le gestionnaire Resized.

```
on resized theObject
  (* Perform operations here after resizing the object. *)
end resized
```

right mouse down

Appelé lorsqu'un Event "right mouse down" (enfoncement du bouton droit de la souris) survient.

Syntaxe

`right mouse down`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference

La référence de l'objet dont le gestionnaire Right Mouse Down est appelé

`[event]` *event* (page 50)

Les informations d'Events de l'Event "right mouse down"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Right Mouse Down à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on right mouse down theObject event theEvent
  (* Add script statements here to handle the right mouse down event. *)
end right mouse down
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "right mouse down", comme l'emplacement de la souris, le nombre de clics, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

right mouse dragged

Appelé lorsqu'un Event "right mouse dragged" (glissement avec le bouton droit de la souris) survient.

Syntaxe

`right mouse dragged`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
: La référence de l'objet dont le gestionnaire Right Mouse Dragged est appelé

`[event]` *event* (page 50)
: Les informations d'Events de l'Event "right mouse dragged"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Right Mouse Dragged à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on right mouse dragged theObject event theEvent
```

```
  (* Add script statements here to handle the right mouse dragged event. *)
end right mouse dragged
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "right mouse dragged", comme l'emplacement de la souris, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

right mouse up

Appelé lorsqu'un Event "right mouse up" (relâchement du bouton droit de la souris) survient.

Syntaxe

`right mouse up`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l'objet dont le gestionnaire Right Mouse Up est appelé

`[event]` *event* (page 50)
Les informations d'Events de l'Event "right mouse up"

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Right Mouse Up à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit.

```
on right mouse up theObject event theEvent
  (* Add script statements here to handle the right mouse up event. *)
end right mouse up
```

Vous pouvez utiliser le paramètre *theEvent* pour obtenir des informations sur l'Event "right mouse up", comme l'emplacement de la souris, et si les touches Commande, Option, Majuscule ou Contrôle ont été enfoncées en même temps. Voir la classe event (page 50) pour des exemples.

scroll wheel

Appelé lorsque le “scroll wheel” se déplace.

Syntaxe

`scroll wheel`	*reference*	obligatoire
`[event]`	*event*	facultatif

Paramètres

reference
La référence de l’objet dont le gestionnaire Scroll Wheel est appelé

`[event]` *event* (page 50)
Les informations d’Events de l’Event “scroll wheel”

Exemples

Le gestionnaire Scroll Wheel suivant répond à un “scroll wheel” en incrémentant ou décrémentant une valeur dans un text field (page 325) basée sur le paramètre event (page 50) transmis. Le gestionnaire awake from nib (page 123) initialise le text field avec une valeur démarrant à 100.

```
on scroll wheel theObject event theEvent
  set theValue to content of theObject as number
  set theValue to theValue + (delta y of theEvent)
  set content of theObject to theValue
end scroll wheel

on awake from nib theObject
  set content of theObject to 100
end awake from nib
```

should close

Appelé lorsqu’un objet supportant ce gestionnaire est sur le point de se fermer.

Ce gestionnaire est supporté par les classes comme window (page 76), panel (page 526) et drawer (page 204). Le gestionnaire peut retourner `false` pour annuler l’opération de fermeture.

Syntaxe

`should close` *reference* obligatoire

Paramètres

reference

La référence de l'objet qui pourrait se fermer

Résultats

boolean

Retourne `true` pour autoriser la fermeture ; `false` pour l'interdire

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Close à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il doit autoriser l'objet à se fermer, puis retournera la valeur appropriée.

```
on should close theObject
  set allowClose to false
  -- Check variable, perform test, or call handler to see if OK to close
  -- If so, set allowClose to true
  return allowClose
end should close
```

should open

Appelé lorsqu'un objet supportant ce gestionnaire (comme une fenêtre, un panel ou un document) est sur le point de s'ouvrir.

Le gestionnaire peut retourner `false` pour annuler l'opération d'ouverture.

Syntaxe

`should open` *reference* obligatoire

Paramètres

reference
: La référence de l'objet qui pourrait s'ouvrir

Résultats

boolean
: Retourne `true` pour autoriser l'ouverture ; `false` pour l'interdire

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Open à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il doit autoriser l'objet à s'ouvrir, puis retournera la valeur appropriée.

```
on should open theObject
  set allowOpen to false
  -- Check variable, perform test, or call handler to see if OK to open
  -- If so, set allowOpen to true
  return allowOpen
end should open
```

should open untitled

Appelé avant qu'un objet supportant ce gestionnaire (généralement l'objet application (page 29)) n'ouvre une fenêtre ou un document "sans titre".

Le gestionnaire peut retourner `false` pour annuler l'opération.

Syntaxe

`should open untitled` *reference* obligatoire

Paramètres

reference
: La référence de l'objet (généralement l'objet application (page 29)) ayant l'option d'ouvrir une fenêtre ou un document "sans titre"

Résultats

boolean

Retournera `true` pour autoriser l'ouverture d'une fenêtre ou d'un document "sans titre" ; `false` pour l'interdire

Exemples

L'exemple suivant de gestionnaire Should Open Untitled vérifie la propriété `allowUntitled`, laquelle est définie ailleurs dans le script, pour déterminer si l'ouverture de l'objet "sans titre" doit être autorisée. Vous pourriez également, à la place, vérifier une propriété ou exécuter une validation dans le gestionnaire lui-même.

```
on should open untitled theObject
  if allowUntitled is equal to true then
    return true
  else
    return false
  end if
end should open untitled
```

should quit

Appelé pour déterminer si l'application doit quitter.

Le gestionnaire peut retourner `false` pour refuser de quitter l'application.

Syntaxe

`should quit` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) qui pourrait quitter

Résultats

boolean

Retournera `true` pour autoriser l'application à quitter ; `false` pour l'interdire

Exemples

L'exemple suivant de gestionnaire Should Quit appelle le gestionnaire `allowQuitting`, écrit par vous, pour déterminer s'il doit autoriser l'application à quitter, puis retournera la valeur appropriée. Vous pourriez également, à la place, vérifier une propriété ou exécuter une validation dans le gestionnaire lui-même.

```
on should quit theObject
  -- Check property, perform test, or call handler to see if OK
  -- to quit
  set allowQuit to allowQuitting(theObject)
  return allowQuit
end should quit
```

should quit after last window closed

Appelé pour déterminer si l'application doit quitter lorsque sa dernière fenêtre est fermée. Le gestionnaire peut retourner `false` pour refuser de quitter l'application.

Syntaxe

`should quit after last window closed` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) qui pourrait quitter lorsque sa dernière fenêtre est fermée

Résultats

boolean

Retournera `true` pour quitter l'application une fois que la dernière fenêtre est fermée ; `false` pour l'interdire

Exemples

L'exemple suivant de gestionnaire Should Quit After Last Window Closed appelle le gestionnaire `shouldQuit`, écrit par vous, pour déterminer s'il

doit autoriser l'application à quitter, puis retournera la valeur appropriée. Vous pourriez également, à la place, vérifier une propriété ou exécuter une validation dans le gestionnaire lui-même.

```
on should quit after last window closed theObject
  -- Check property, perform test, or call handler to see if OK
  -- to quit after last window closed
  set allowQuit to shouldQuit(theObject)
  return allowQuit
end should quit after last window closed
```

should zoom

Appelé pour déterminer si une fenêtre doit être agrandie.

Le gestionnaire peut examiner les limites de zoom et retourner `false` s'il refuse l'agrandissement ou `true` s'il l'autorise. En cas d'absence de gestionnaire ou si vous fournissez un gestionnaire mais qui ne retourne pas de valeur, par défaut l'agrandissement sera autorisé.

Si vous souhaitez pouvoir contrôler les limites du zoom, utilisez le gestionnaire will zoom (page 169). Si vous voulez accéder aux limites dans le gestionnaire Will Zoom, vous devrez implémenter Should Zoom et enregistrer la valeur du paramètre *proposed bounds* afin de pouvoir l'utiliser plus tard avec le gestionnaire will zoom (page 169).

Syntaxe

`should zoom`	*reference*	obligatoire
`[proposed bounds]`	*bounding rectangle*	facultatif

Paramètres

reference
: La référence de l'objet window (page 76) qui pourrait être agrandie

`[proposed bounds]` *bounding rectangle*
: Les limites demandées pour l'objet devant être agrandie ; une liste de quatre nombres {gauche, bas, droite, haut} ; voir la propriété *bounds* de la classe window (page 76) pour des informations sur le système des coordonnées

Résultats

boolean

Retournera `false` pour interdire le zoom ou `true` pour l'autoriser. Si vous implémentez ce gestionnaire, vous devrez toujours retourner une valeur booléenne (`true` ou `false`)

Exemples

L'exemple suivant de gestionnaire Should Zoom appelle le gestionnaire `isZoomable`, écrit par vous, pour déterminer s'il doit autoriser l'agrandissement, puis retournera la valeur appropriée. Si vous voulez accéder aux limites demandées dans le gestionnaire will zoom (page 169), vous devrez utiliser Should Zoom pour enregistrer la valeur du paramètre *proposed bounds* pour un usage ultérieur.

```
on should zoom theObject
  -- Check property, perform test, or call handler to see if OK to edit
  set allowZooming to isZoomable(theObject)
  return allowZooming
end should zoom
```

Version

Le paramètre `[proposed bounds]` est apparu avec la version 1.2 d'AppleScript Studio.

Dans le dictionnaire d'AppleScript Studio version 1.3 (`AppleScriptKit.asdictionary`), il est indiqué que le type du paramètre *proposed bounds* est *point*. Le type correct est *bounding rectangle*, une liste de quatre nombres entiers {gauche, bas, droite, haut}.

shown

Appelé après qu'un objet application (page 29) soit montré.

Syntaxe

`shown` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) ayant été montré

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Shown à un objet application (page 29), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois l'objet application montré.

```
on shown theObject
  (* Perform operations here after the object is shown. *)
end shown
```

updated

Appelé après qu'un objet soit mis à jour.

Syntaxe

`updated` *reference* obligatoire

Paramètres

reference

La référence de l'objet ayant été mis à jour

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Updated à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois la mise à jour faite.

```
on updated theObject
  (* Perform operations here after the object is updated. *)
end updated
```

was hidden

Appelé après que l'objet application (page 29) ait été caché, soit avec le menu "Masquer l'Application" du menu "Application" (ici Application remplace le véritable nom de l'application) ou en appuyant sur les touches Cmd + H.

Vous ne pouvez pas connecter ce gestionnaire à un objet window (page 76), bien que vous puissiez utiliser la commande hide (page 99) pour cacher une fenêtre. Votre application peut explicitement appeler des commandes comme Hide, mais lorsque vous connecterez des gestionnaires comme Was Hidden aux objets, les gestionnaires seront appelés par AppleScript Studio au moment approprié.

Syntaxe

`was hidden` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) ayant été caché

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Was Hidden à un objet application (page 29), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois l'objet application caché.

```
on was hidden theObject
  (* Perform operations here after the object was hidden. *)
end was hidden
```

was miniaturized

Appelé après qu'une fenêtre ait été réduite.

Le gestionnaire peut exécuter toute opération ayant besoin de la réduction.

Vous devriez utiliser miniaturized (page 136), plutôt que Was Miniaturized.

Syntaxe

`was miniaturized` *reference* obligatoire

Paramètres

reference

La référence de l'objet window (page 76) ayant été réduit

will become active

Appelé lorsque l'objet est sur le point de devenir actif.

Le gestionnaire ne peut pas annuler l'activation, mais peut la préparer.

L'application, lors de sa phase de lancement, appelle certains gestionnaires, s'ils sont présents, en respectant un ordre de priorité. Le gestionnaire Will Become Active fait partie de ces gestionnaires, la liste est disponible dans la description du gestionnaire awake from nib (page 123).

Syntaxe

`will become active` *reference* obligatoire

Paramètres

reference

La référence de l'objet devenant l'objet actif

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Become Active à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer l'activation.

```
on will become active theObject
  (* Perform operations here before becoming active. *)
end will become active
```

will close

Appelé lorsque l'objet est sur le point de se fermer.

Le gestionnaire ne peut pas annuler l'opération de fermeture, mais peut la préparer. Les classes comme window (page 76), panel (page 526) et drawer (page 204) supportent le gestionnaire Will Close.

Syntaxe

`will close` *reference* obligatoire

Paramètres

reference

La référence de la fenêtre, du tiroir ou d'un autre objet sur le point de se fermer

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Close à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer la fermeture. Par exemple, vous pourriez vouloir mettre à jour un objet contrôleur avec les informations des objets text field (page 325) d'une fenêtre. Notez qu'avant vous devrez valider le contenu des objets text field dans un gestionnaire should close (page 151), au cas où la fenêtre refuserait de se fermer si un des champs contient des données non valides.

```
on will close theObject
  (* Perform operations here before closing. *)
end will close
```

will finish launching

Appelé lorsque l'objet application (page 29) est sur le point de finir son lancement.

Le gestionnaire ne peut pas annuler le lancement, mais peut le préparer.

L'application, lors de sa phase de lancement, appelle certains gestionnaires, s'ils sont présents, en respectant un ordre de priorité. Le gestionnaire

Will Finish Launching fait partie de ces gestionnaires, la liste est disponible dans la description du gestionnaire awake from nib (page 123).

Syntaxe

`will finish launching` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) qui va finir son lancement

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Finish Launching à un objet, AppleScript Studio ajoute automatiquement, au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération une fois que le fichier .nib principal ait été chargé, mais avant la fin du démarrage de l'application. Par exemple, le script suivant rend visible la fenêtre du fichier .nib principal.

```
on will finish launching theObject
  (* Perform operations here before completion of launching. *)
  set visible of (window of theObject) to true
end will finish launching
```

Vous pourriez aussi utiliser le gestionnaire Will Finish Launching pour vérifier la présence de la version minimale du runtime d'AppleScript Studio requise par votre application. Pour des informations sur les versions du runtime, voir "Information sur les versions" (page 8).

Le prochain exemple montre comment le gestionnaire pourrait vérifier la version requise d'AppleScript Studio. Si la version est disponible, le gestionnaire répertorie un "drag type", lequel est uniquement supporté depuis la version 1.2 d'AppleScript Studio. Si la version 1.2 n'est pas disponible, l'application affiche un message et puis quitte. Notez que le gestionnaire ne vérifie pas directement le numéro de version d'AppleScript Studio. Au lieu de cela, il vérifie la version correspondante d'AppleScript Studio, comme montré dans le tableau 1.1 (page 9).

```
on will finish launching theObject
  if AppleScript's version as string greater than "1.9" then
```

```
    tell window 1 to register drag types {"file names"}
  else
    display dialog "This application requires AppleScript Studio 1.2 or later."
    quit
  end if
end will finish launching
```

will hide

Appelé lorsque que l'objet application (page 29) est sur le point d'être caché, soit par le menu "Masquer l'application" du menu "application" (ici application remplace le nom de l'application), soit en appuyant sur les touches Cmd + H.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer.

Vous ne pouvez pas connecter ce gestionnaire à un objet window (page 76), bien que vous puissiez utiliser la commande hide (page 99) pour cacher une fenêtre. Votre application peut explicitement appeler des commandes comme Hide, mais lorsque vous connecterez des gestionnaires comme Will Hide aux objets, les gestionnaires seront appelés par AppleScript Studio au moment approprié.

Syntaxe

`will hide` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29) qui va être caché

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Hide à un objet application (page 29), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le masquage de l'application.

```
on will hide theObject
  (* Perform operations here before the application hides. *)
end will hide
```

will miniaturize

Appelé lorsqu'un objet supportant ce gestionnaire (comme une fenêtre ou un panel) est sur le point d'être réduit.

Le gestionnaire ne peut pas annuler l'opération de réduction, mais peut la préparer.

Syntaxe

`will miniaturize` *reference* obligatoire

Paramètres

reference

La référence de l'objet qui va être réduit

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Miniaturize à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer la réduction.

```
on will miniaturize theObject
  (* Perform operations here before the object miniaturizes. *)
end will miniaturize
```

will move

Appelé lorsqu'un objet est sur le point d'être déplacé.

Le gestionnaire ne peut pas annuler l'opération de déplacement, mais peut la préparer.

Syntaxe

`will move` *reference* obligatoire

Paramètres

reference

La référence de l'objet qui va être déplacé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Move à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le déplacement.

```
on will move theObject
  (* Perform operations here before the object moves. *)
end will move
```

will open

Appelé lorsqu'un objet supportant ce gestionnaire (comme une fenêtre ou un panel) est sur le point de s'ouvrir.

Le gestionnaire ne peut pas annuler l'opération d'ouverture, mais peut la préparer.

Syntaxe

`will open` *reference* obligatoire

Paramètres

reference

La référence de l'objet qui va s'ouvrir, comme un objet window (page 76) ou panel (page 526)

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Open à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer l'ouverture.

```
on will open theObject
  (* Perform operations here before the object opens. *)
end will open
```

Version

Dans les versions antérieures à la version 1.2 d'AppleScript Studio, vous pouviez connecter un gestionnaire Will Open à un objet document (page 457).

Dans les versions antérieures à la version 1.2 d'AppleScript Studio, un gestionnaire Will Open n'était appelé que lorsque la fenêtre était chargée depuis son fichier .nib, pas lorsqu'elle était ouverte. Pour simuler cette caractéristique dans la version 1.2, vous pouvez remplacer les appels au gestionnaire Will Open par des appels au gestionnaire awake from nib (page 123).

`will quit`

Appelé lorsque l'objet application (page 29) est sur le point de quitter.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer.

Syntaxe

`will quit` *reference* obligatoire

Paramètres

reference

La référence de l'objet application (page 29)

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Quit à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer l'application à quitter.

```
on will quit theObject
  (* Perform operations here before the application quits. *)
end will quit
```

will resign active

Appelé lorsqu'un objet est sur le point d'abandonner son état actif.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer. Il n'existe pas de gestionnaire autorisant l'application à refuser l'abandon de son état actif.

Syntaxe

`will resign active`	*reference*	obligatoire

Paramètres

reference
: La référence de l'objet application (page 29) qui va abandonner son état actif

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Resign Active à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer l'abandon de l'état actif.

```
on will resign active theObject
  (* Perform operations here before the object resigns its active state. *)
end will resign active
```

will resize

Appelé lorsqu'un objet window (page 76) est sur le point d'être redimensionné.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer, et peut retourner une taille différente pour spécifier la nouvelle taille.

Syntaxe

`will resize`	*reference*	obligatoire
`[proposed size]`	*point*	facultatif

Paramètres

reference

La référence de l'objet window (page 76) qui va être redimensionné

`[proposed size]` *point*

La taille demandée de la fenêtre, consistant en une liste de deux nombres entiers {horizontal, vertical} ; voir la propriété *bounds* de la classe window (page 76) pour des informations sur le système des coordonnées

Résultats

point

Vous pouvez retourner un "point" différent pour spécifier la taille à laquelle la fenêtre sera redimensionnée. Si vous ne retournez pas un "point", la valeur qui était transmise dans le paramètre *proposed size* est utilisée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Resize à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le redimensionnement, et vous pouvez retourner une valeur de taille pour spécifier la nouvelle taille. Si vous avez besoin de connaître l'ancienne taille dans un gestionnaire resized (page 147) (lequel est appelé après le redimensionnement), vous pouvez utiliser le gestionnaire Will Resize pour enregistrer la taille courante.

```
on will resize theObject
  (* Perform operations here before the object resizes.
  Return a point to specify a different size.
  For example: *)
  return {200, 560}
end will resize
```

Version

Le paramètre `proposed size` est apparu avec la version 1.2 d'AppleScript Studio.

will show

Appelé lorsqu'un objet est sur le point d'être montré.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer.

Syntaxe

`will show` *reference* obligatoire

Paramètres

reference
La référence de l'objet qui va être montré

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Show à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer "l'exposition".

```
on will show theObject
  (* Perform operations here before the object is shown. *)
end will show
```

will zoom

Appelé lorsqu'un objet window (page 76) est sur le point d'être agrandi.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer.

Syntaxe

`will zoom`	*reference*	obligatoire
`[screen bounds]`	*bounding rectangle*	facultatif

Paramètres

reference
La référence de l'objet window (page 76) qui va être agrandi

`[screen bounds]` *bounding rectangle*
Une liste de quatre nombres {gauche, bas, droite, haut} spécifiant les

limites de l'écran contenant la partie la plus large de la fenêtre (moins le Dock et la barre de menus, s'il le faut) ; notez qu'il ne s'agit pas de limites demandées (ou de limites pour lesquelles la fenêtre sera agrandie) — voir la propriété *bounds* de la classe window (page 76) pour des informations sur le système des coordonnées

Résultats

bounding rectangle

Vous pouvez retourner un "bounding rectangle" (une liste de quatre nombres {gauche, bas, droite, haut}) pour spécifier les limites de la fenêtre agrandie. Si vous ne retournez pas un rectangle, la valeur qui était à l'origine transmise avec le paramètre `screen bounds` est utilisée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Zoom à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre *theObject* référence l'objet qui va être agrandi. Vous pouvez utiliser le paramètre *screen bounds*, ainsi que les limites de la fenêtre elle-même, et peut-être les limites demandées pour l'opération de zoom (si vous les avez enregistrées dans le gestionnaire should zoom (page 156)) pour déterminer s'il faut modifier les limites pour lesquelles la fenêtre va être agrandie.

Le gestionnaire suivant modifie les limites de la fenêtre agrandie à 70 pixels depuis le côté gauche de l'écran. Il fait cela en obtenant les limites courantes de la fenêtre, en réglant la valeur de son élément gauche à 70, et en retournant ces limites pour le zoom.

```
on will zoom theObject screen bounds screenBounds
  (* Perform operations here before the object is zoomed. *)
  set theBounds to bounds of theObject
  set item 1 of theBounds to 70
  return theBounds
end will zoom
```

Version

L'Event Will Zoom est apparu avec la version 1.2 d'AppleScript Studio.

zoomed

Appelé après qu'un objet (généralement une fenêtre ou un panel) soit agrandi.

Le gestionnaire ne peut pas annuler l'opération, mais peut la préparer.

Syntaxe

`zoomed` *reference* obligatoire

Paramètres

reference

La référence de l'objet ayant été agrandi

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Zoomed à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le zoom.

```
on zoomed theObject
  (* Perform operations here after the object zooms. *)
end zoomed
```

Chapitre 4

Énumérations

La suite Application fournit les énumérations suivantes, lesquelles définissent les constantes utilisables dans les applications AppleScript Studio.

Important
Ces énumérations sont disponibles pour toutes les classes des suites d'AppleScript Studio.

alert return values 174
alert type 174
bezel style 175
border type 175
box type 176
button type 177
cell image position 177
cell state value 178
cell type 179
color panel mode 179
control size 180
control tint 180
drawer state 181
event type 181
go to 183
image alignment 183
image frame style 184
image scaling 184

matrix mode 185
quicktime movie loop mode 185
rectangle edge 186
scroll to location 186
sort case sensitivity 187
sort order 187
sort type 188
tab state 188
tab view type 188
text alignment 189
tick mark position 190
title position 190

alert return values

Non utilisées dans la version 1.3 d'AppleScript Studio.

Constantes

alternate return
Le bouton sur deux retourné

default return
Le bouton retourné par défaut

error return
L'erreur retournée

other return
L'autre bouton retourné

alert type

Spécifie le niveau d'alerte.

Vous transmettrez une de ces valeurs à la commande display alert (page 540).

Constantes

critical
Alerte critique

informational
Alerte informative

warning
Avertissement

bezel style

Spécifie l'apparence du contour.

La propriété *bezel style* d'un objet button (page 254) sera réglée avec l'une des valeurs listées ci-dessous.

Constantes

circular bezel
Contour circulaire (pour un bouton rond)

regular square bezel
Contour carré régulier (pour un bouton carré)

rounded bezel
Contour arrondi (pour un bouton avec un bord arrondi)

shadowless square bezel
Contour carré ombré (pour un bouton carré)

thick square bezel
Contour carré épais (pour un bouton carré)

thicker square bezel
Contour carré plus épais (pour un bouton carré)

border type

Spécifie le type de contour.

Les objets comme box (page 197) et scroll view (page 213) possèdent des propriétés de contour. Border Type peut interagir avec box type (page 176).

Constantes

bezel border
: Contour “collerette”

groove border
: Contour rainuré (lorsqu'elle est combinée avec la valeur *separator type* de box type (page 176), on obtient une ligne de séparation)

line border
: Contour “ligne” (dépendante de box type (page 176), produit des contours à facettes)

no border
: Sans contour

box type

Spécifie le modèle de boîte.

La propriété *box type* d'un objet box (page 197) sera réglée avec l'une des valeurs listées ci-dessous. Le modèle de boîte choisi peut interagir avec border type (page 175). Vous pourrez trouver des illustrations des différents modèles de boîtes, y compris des “groups boxes”, dans le guide “*Aqua Human Interface Guidelines*”, disponible dans l'aide de Xcode.

Constantes

old style type
: Vieux modèle de boîte (si la propriété *border type* est réglée sur *bezel border*, ombrée avec une bordure en haut et à gauche ; si la propriété *border type* est réglée sur *line border*, claire avec une bordure de chaque côté)

primary type
: Modèle primaire (terminologie d'un modèle de “group box” décrit dans une ancienne guideline ; la guideline de l'interface Aqua désapprouve l'utilisation des “groups boxes”)

secondary type
: Modèle secondaire (terminologie d'un modèle de “group box” décrit dans une ancienne guideline ; la guideline de l'interface Aqua désapprouve l'utilisation des “groups boxes”)

separator type
: Modèle séparateur (lorsque ce modèle est combiné avec la constante *groove border* de border type (page 175), il produit une ligne de séparation)

button type

Spécifie le modèle de bouton, lequel peut affecter à la fois l'aspect du bouton et son comportement.

La propriété *button type* d'un objet button (page 254) sera réglée avec l'une des valeurs listées ci-dessous.

Constantes

momentary change button
: Momentanément modifié

momentary light button
: Momentanément allumé

momentary push in button
: Momentanément enfoncé

on off button
: Bouton on / off

push on off button
: Bouton poussoir

radio button
: Bouton radio

switch button
: Bouton case à cocher

toggle button
: Bouton bascule

cell image position

Spécifie la position de l'image dans une cellule.

Les objets comme button (page 254) et cell (page 264) possèdent tous les deux les propriétés *image* et *image position*.

Constantes

image above
Image au-dessus

image below
Image en-dessous

image left
Image à gauche

image only
Uniquement montrer l'image

image overlaps
Chevauchement de l'image

image right
Image à droite

no image
Aucune image

`cell state value`

Spécifie l'état d'une cellule.

La classe cell (page 264) fournit à la fois une propriété *state* et une propriété *support mixed state*. Ses sous-classes peuvent supporter deux états (activé ou désactivé) ou trois états (activé, désactivé ou mixte).Un état mixte est utile pour une case à cocher ou un bouton radio, cela permet de refléter l'état d'une caractéristique qui est vraie pour certains éléments. Par exemple, une case à cocher "italique" sera activée si tout le texte de la sélection courante est en italique, désactivée si tous les caractères ne sont pas en italique, et mixte si seulement certains caractères sont en italique.

Constantes

mixed state
État mixte

off state
État désactivé

on state
État activé

cell type

Spécifie le type de cellule.

La classe cell (page 264) fournit une propriété *cell type* pour ses sous-classes comme image cell (page 284) et text field cell (page 330).

Constantes

image cell type
Cellule prévue pour une image

null cell type
Cellule vide

text cell type
Cellule prévue pour du texte

color panel mode

Spécifie comment un utilisateur peut sélectionner une couleur dans un objet color-panel (page 514).

Ce masque est réglé avant que vous initializiez une nouvelle instance de color-panel (page 514). Vous pouvez régler la propriété *color mode* d'un objet color-panel (page 514) avec n'importe laquelle des valeurs listées ci-dessous.

Bien que les nuanciers peuvent supporter plusieurs modes, lorsque vous obtenez ou que vous réglez dans un objet AppleScript Studio une propriété de couleur, celle-ci sera au format RVB. Une couleur RVB est représentée par une liste à trois éléments, chaque élément représentant un des composants de la couleur. Par exemple, la couleur bleue sera représentée par la valeur {0,0,65535}.

Constantes

cmyk mode
Mode CMJN (Cyan, Magenta, Jaune et Noir)

color list mode
Mode liste de couleurs

color wheel mode
Mode roue de couleurs

custom palette mode
: Mode couleurs personnalisées

gray mode
: Mode niveau de gris

hsb mode
: Mode couleur HSB

rgb mode
: Mode couleur RVB (Rouge, Vert et Bleu)

control size

Spécifie la taille des contrôles.

Les classes comme cell (page 264), progress indicator (page 306) et tab view (page 221) possèdent une propriété *control size* qui pourra être réglée avec l'une des valeurs listées ci-dessous. *small size*, que vous pouvez régler dans Interface Builder, fournit les plus petites versions de ces éléments d'interface, convenant aux fenêtres ou aux panels plus petits.

Constantes

regular size
: Taille normale

small size
: Taille réduite

control tint

Spécifie le thème des contrôles.

Les classes comme cell (page 264), progress indicator (page 306) et tab view (page 221) possèdent une propriété *control tint* qui pourra être réglée avec l'une des valeurs listées ci-dessous. Par exemple, lorsque la propriété *control tint* est réglée sur *default tint*, le thème en cours est Aqua.

Constantes

clear tint
: Thème transparent

default tint
: Thème par défaut, Aqua

drawer state

Spécifie l'état d'un objet drawer (page 204) (tiroir).

Pour plus d'informations, voir la classe drawer (page 204).

Constantes

drawer closed
: Le tiroir est fermé

drawer closing
: Le tiroir se ferme

drawer opened
: Le tiroir est ouvert

drawer opening
: Le tiroir s'ouvre

event type

Spécifie le type d'Events.

Pour plus d'informations, voir la classe event (page 50).

Constantes

appkit defined type
: Event défini par AppKit (Cocoa se compose de deux frameworks, AppKit et Foundation)

application defined type
: Event défini par l'objet application (page 29)

cursor update type
: Event mise à jour du curseur

flags changed type
: Event touche de fonction modifiée

key down type
: Event enfoncement d'une touche du clavier

key up type
: Event relâchement de la touche du clavier

left mouse down type
: Event enfoncement du bouton gauche de la souris

left mouse dragged type
: Event glissé avec le bouton gauche de la souris

left mouse up type
: Event relâchement du bouton gauche de la souris

mouse entered type
: Event entrée du curseur de la souris

mouse exited type
: Event sortie du curseur de la souris

mouse moved type
: Event déplacement du curseur de la souris

other mouse down type
: Event enfoncement d'un autre bouton de la souris

other mouse dragged type
: Event glissé avec un autre bouton de la souris

other mouse up type
: Event relâchement d'un autre bouton de la souris

periodic type
: Event périodique, comme un Event idle ou une minuterie

right mouse down type
: Event enfoncement du bouton droit de la souris

right mouse dragged type
: Event glissé avec le bouton droit de la souris

right mouse up type
: Event relâchement du bouton droit de la souris

scroll wheel type
: Event défilement avec la molette

system defined type
Event défini par le système

go to

Spécifie un emplacement dans un film.

Vous transmettrez une de ces valeurs avec la commande go (page 334). Pour d'autres informations, voir la classe movie view (page 296).

Constantes

beginning frame
Au début

end frame
À la fin

poster frame
Un "poster frame"

image alignment

Spécifie l'alignement d'une image.

Les classes comme image cell (page 284) et image view (page 285) possèdent une propriété *image alignment*.

Constantes

bottom alignment
Alignement sur le bas

bottom left alignment
Alignement sur le coin inférieur gauche

bottom right alignment
Alignement sur le coin inférieur droit

center alignment
Alignement au centre

left alignment
Alignement à gauche

right alignment
Alignement à droite

top alignment
Alignement sur le haut

top left alignment
Alignement sur le coin supérieur gauche

top right alignment
Alignement sur le coin supérieur droit

image frame style

Spécifie le style du cadre d'une image.

Les classes comme image cell (page 284) et image view (page 285) possèdent une propriété *frame style*.

Constantes

button frame
Cadre de bouton

gray bezel frame
Cadre gris

groove frame
Cadre rainuré

no frame
Pas de cadre

photo frame
Cadre de photo

image scaling

Spécifie la mise à l'échelle d'une image.

Les classes comme image cell (page 284) et image view (page 285) possèdent une propriété *image scaling*.

Constantes

no scalling
: Pas de mise à l'échelle

scale proportionally
: Mise à l'échelle proportionnelle

scale to fit
: Mise à l'échelle pour le mieux

`matrix mode`

Spécifie les modalités d'une structure ou d'une matrice.

Une matrice est un groupe de cellules qui fonctionnent en concert, comme des boutons radio.

Constantes

highlight mode
: Une cellule est "illuminée" avant qu'elle ne soit invitée à traquer la souris, puis "éteinte" une fois fini

list mode
: Les cellules sont "illuminées", mais elles ne traquent pas la souris

radio mode
: Sélection d'une cellule à la fois. Lorsqu'une cellule est sélectionnée, la cellule précédente est désélectionnée

track mode
: Les cellules sont invitées à traquer la souris chaque fois que celle-ci est à l'intérieur des limites. Aucun "illuminement" n'est exécuté

`quicktime movie loop mode`

Spécifie le mode de lecture d'un film.

Vous pouvez obtenir ou régler la propriété *loop mode* d'un objet movie view (page 296).

Constantes

looping back and forth playback
: La lecture se fait en boucle et en va-et-vient, d'avant en arrière et d'arrière en avant.

looping playback
: Lecture en boucle (la lecture recommence au début une fois le film fini)

normal playback
: Lecture normale (la lecture s'arrête une fois le film fini)

`rectangle edge`

Spécifie le côté sur lequel un tiroir pourrait s'ouvrir ou un menu pop-up se dérouler si l'espace est limité. Par exemple, vous utiliserez ces valeurs pour obtenir ou régler la propriété *edge* d'un objet drawer (page 204) ou *preferred edge* pour un objet popup button (page 301).

Constantes

bottom edge
: Bord inférieur

left edge
: Côté gauche

right edge
: Côté droit

top edge
: Bord supérieur

`scroll to location`

Spécifie la partie de la fenêtre à atteindre en cas de défilement.

À utiliser avec la commande scroll (page 339). Toutefois, cette commande n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Constantes

bottom
: Inférieure

top
: Supérieure

visible
: Visible

`sort case sensitivity`

Spécifie si un tri doit tenir compte ou non de la casse, comme lors du tri des colonnes d'une Data Source.

Constantes

case insensitive
: Tri des données sans tenir compte de la casse

case sensitive
: Tri des données compte tenu de la casse

Version

`sort case sensitivity` est apparu avec la version 1.2 d'AppleScript Studio.

`sort order`

Spécifie l'ordre dans lequel se fait le tri, comme lors du tri des colonnes d'une Data Source.

Constantes

ascending
: Tri croissant

descending
: Tri décroissant

Version

`sort order` est apparu avec la version 1.2 d'AppleScript Studio.

`sort type`

Spécifie le type de tri, comme lors du tri des colonnes d'une Data Source.

Constantes

alphabetical
: Tri alphabétique

numerical
: Tri numérique

Version

`sort type` est apparu avec la version 1.2 d'AppleScript Studio.

`tab state`

Spécifie l'état d'un onglet d'un objet tab view item (page 227).

La propriété *tab state* d'un objet tab view item est en lecture uniquement.

Constantes

background
: Onglet en arrière-plan

pressed
: Onglet appuyé

selected
: Onglet sélectionné

`tab view type`

Spécifie le type d'un objet tab view (page 221), par la position et le format des onglets.

Constantes

bottom tabs bezel border
: Étiquettes en bas, contour “collerette”

left tabs bezel border
: Étiquettes à gauche, contour “collerette”

no tabs bezel border
: Sans étiquettes, contour “collerette”

no tabs line border
: Sans étiquettes, contour “ligne”

no tabs no border
: Sans étiquettes, sans contour

right tabs bezel border
: Étiquettes à droite, contour “collerette”

top tabs bezel border
: Étiquettes en haut, contour “collerette”

Version

bottom tabs bezel border, *left tabs bezel border*, et *right tabs bezel border* fonctionnent uniquement avec la version de Cocoa livrée avec Mac OS X 10.2.

`text alignment`

Spécifie l'alignement du texte.

Voir la propriété *alignment* de la classe text (page 569).

Constantes

center text alignment
: Texte centré

justified text alignment
: Texte justifié

left text alignment
: Texte aligné à gauche

natural text alignment
: Alignement du texte sur l'alignement par défaut

right text alignment
: Texte aligné à droite

tick mark position

Spécifie la position de la graduation.

Voir la propriété *tick mark position* de la classe slider (page 315).

Constantes

tick mark above
: Graduation au-dessus

tick mark below
: Graduation en-dessous

tick mark left
: Graduation à gauche

tick mark right
: Graduation à droite

title position

Spécifie la position du titre.

Voir la propriété *title position* de la classe box (page 197).

Constantes

above bottom
: Titre en bas au-dessus de la ligne de base

above top
: Titre en haut au-dessus de la ligne de base

at bottom
: Titre en bas

at top
: Titre en haut

below bottom
: Titre en bas en-dessous de la ligne de base

below top
: Titre en haut en-dessous de la ligne de base

no title
: Sans titre

Troisième partie

Container View Suite

Cette partie décrit la terminologie de la suite Container View d'AppleScript Studio.

La suite Container View définit la classe view (page 229). La plupart des classes de la suite Container View hérite de la classe responder (page 68), soit directement, soit par l'intermédiaire de la classe view. La suite Container View définit aussi les Events fonctionnant avec les containers views et les views qu'ils contiennent.

Les classes, commandes et Events de la suite Container View sont décrits dans les chapitres suivants :

Classes 197
Commandes 239
Events 243

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Container View contient les classes suivantes :

box 197
clip view 202
drawer 204
scroll view 213
split view 218
tab view 221
tab view item 227
view 229

box

Pluriel : `boxes`
Hérite de : `view` (page 229)
Classe Cocoa : NSBox

Une view basique pouvant avoir son propre contour et son propre titre.

Vous pouvez utiliser un objet box pour visuellement regrouper d'autres views ou pour servir comme simples séparateurs.

L'illustration 3.1 montre plusieurs objets box, comprenant une boîte vide, une boîte contenant deux boutons radio et des boîtes utilisées comme séparateur vertical et horizontal. Dans Interface Builder, vous trouverez les "separator boxes" dans le panneau "Cocoa-Controls" (parmi divers boutons

et champs texte), tandis que les “containers boxes” sont dans le panneau “Cocoa-Containers” (parmi divers objets views).

Voir la classe scroll view (page 213) pour plus d’informations sur la manière de mettre les objets dans les sous-views d’un objet box ou d’autres views dans Interface Builder.

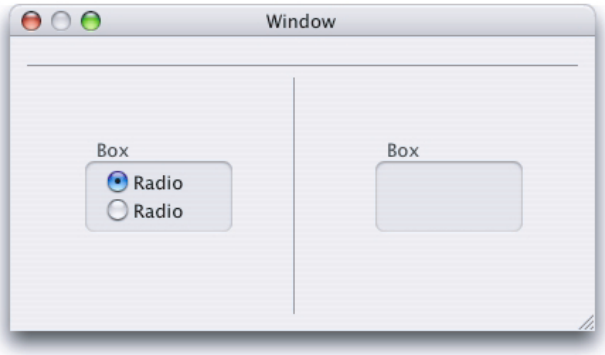

FIG. 3.1 - *Des “boxes”, utilisés comme séparateur horizontal et vertical*

Propriétés des objets de la classe Box

En plus des propriétés qu’il hérite de view (page 229), un objet box possède ces propriétés :

border rect

Accès : lecture uniquement
Classe : *bounding rectangle*
Les limites du contour ; une liste de quatre nombres, {gauche, bas, droite, haut} ; les objets box ont leur propre système de coordonnées, aussi les valeurs de {gauche,bas} seront toujours égales à {0,0} ; voir la propriété *bounds* de la classe window (page 76) pour plus d’informations sur le système des coordonnées

border type

Accès : lecture / écriture
Classe : *une des constantes* de border type (page 175)
Le type de contour

box type

Accès : lecture / écriture
Classe : *une des constantes* de box type (page 176)

Le type d'objet box

content view

Accès : lecture / écriture
Classe : *view* (page 229)
Le contenu visuel de l'objet box, lequel contient toutes ses sous-views ; pour des informations de même nature, voir la propriété *content view* de la classe window (page 76)

content view margins

Accès : lecture / écriture
Classe : *point*
Les marges du content view, calculées depuis le bord de l'objet box ; une liste de deux nombres {gauche, bas} ; par défaut elle est égale à {5.0, 5.0} ; afin de pouvoir modifier les réglages de cette propriété, vous devrez utiliser la commande call method (page 94), comme dans l'exemple suivant :

```
tell box 1
  set content view margins to {2.5, 2.5}
  call method "sizeToFit"
end tell
```

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de l'objet box

title cell

Accès : lecture uniquement
Classe : *cell* (page 264)
La cellule du titre

title font

Accès : lecture / écriture
Classe : *font* (page 54)
Non supportée dans la version 1.3 d'AppleScript Studio ; la police de la cellule du titre

title position

Accès : lecture / écriture
Classe : *une des constantes* de title position (page 190)

La position du titre

title rect

Accès : lecture uniquement
Classe : *bounding rectangle*
Les limites du titre ; une liste de quatre nombres, {gauche, bas, droite, haut} ; mis au point à l'intérieur du système de coordonnées de l'objet box ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

Éléments des objets de la classe Box

Un objet box peut uniquement contenir les éléments qu'il hérite de view (page 229).

Events supportés par les objets de la classe Box

Un objet box supporte les gestionnaires répondant aux Events suivants :

Glisser-déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)

`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Pour une fenêtre “main” contenant un objet box avec plusieurs champs texte, vous pourriez régler le texte d’un des champs avec l’instruction suivante :

```
set contents of text field "company" of box "info" of window "main"
  to "Acme Nuts and Bolts, Ltd."
```

Pour accéder aux différentes propriétés du même objet box, vous pourriez écrire :

```
tell window "main"
  tell box "info"
    set boxTitle to title
    set boxType to box type
    -- etc...
  end tell
end tell
```

Version

Le support des Events de glisser-déposer est apparu avec la version 1.2 d’AppleScript Studio.

La propriété *title font* de cette classe n’est pas supportée dans la version 1.3 d’AppleScript Studio.

clip view

Pluriel : clip views
Hérite de : view (page 229)
Classe Cocoa : NSClipView

Une view contenant et faisant défiler la view affichée par un objet scroll view (page 213).

Vous n'avez normalement pas besoin de scripter l'objet clip view, car la classe Scroll View gère la plupart des détails de défilement lorsque la taille du document view change ou si la position le requiert. La classe Scroll View fournit aussi l'accès à la plupart des mêmes propriétés listées par la classe Clip View.

Propriétés des objets de la classe Clip View

En plus des propriétés qu'il hérite de view (page 229), un objet clip view possède ces propriétés :

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du fond du clip view : une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur bleue peut être représentée par {0, 0, 65535} ; par défaut, la liste est égale à {65535, 65535, 65535} ou à la couleur blanche

content view

Accès : lecture / écriture
Classe : *view* (page 229)
Le contenu visuel du clip view, lequel contient toutes ses sous-views ; pour des informations de même nature, voir la propriété *content view* de la classe window (page 76)

copies on scroll

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le contenu de la view soit copié lorsqu'il est déroulé ?

document rect

Accès : lecture uniquement
Classe : *bounding rectangle*
Les limites du document view dans le clip view ; une liste de quatre nombres, {gauche, bas, droite, haut} ; réglées dans le système des coordonnées du clip view ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

document view

Accès : lecture / écriture
Classe : *view* (page 229)
Le subview principal du clip view ; par exemple, un objet table view (page 403) ou text view (page 569)

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet clip view dessine son fond ?

visible document rect

Accès : lecture uniquement
Classe : *bounding rectangle*
Les limites visibles du document view ; une liste de quatre nombres, {gauche, bas, droite, haut} ; réglées dans le système des coordonnées du clip view ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

Éléments des objets de la classe Clip View

Un objet clip view peut uniquement contenir les éléments qu'il hérite de view (page 229).

Events supportés par les objets de la classe Clip View

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

drawer

Pluriel : drawers
Hérite de : responder (page 68)
Classe Cocoa : NSDrawer

Un élément d'interface contenant et affichant des objets view (page 229).

Les tiroirs (drawers en anglais) contiennent généralement des objets text field (page 325), scroll view (page 213), browser (page 363) et d'autres objets basés sur les classes héritant de la classe view (page 229). L'illustration 3.2 montre un tiroir qui contient plusieurs objets d'interface.

Un objet drawer est associé à un objet window (page 76), appelé son parent, et peut uniquement apparaître lorsque son parent est visible à l'écran. Un objet drawer ne peut pas être déplacé ou commandé indépendamment de la fenêtre associée, mais il est par contre attaché à un des bords de son parent et se déplace le long de celui-ci.

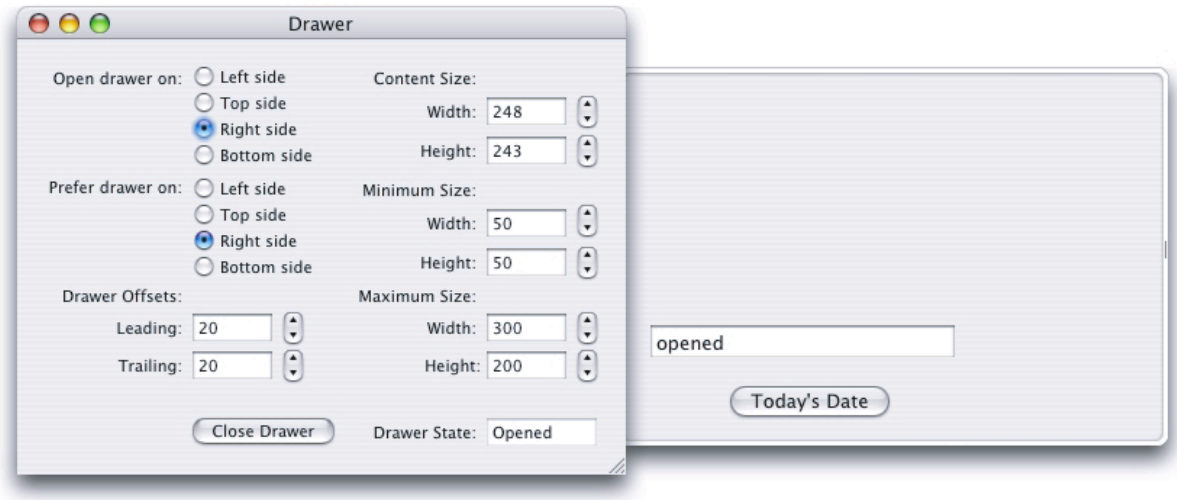

FIG. 3.2 - *Une fenêtre avec un tiroir ouvert (extrait de l'application "Drawer")*

Pour ajouter un tiroir à votre application AppleScript Studio, vous le glisserez-déposerez depuis le panneau "Cocoa-Windows", visible dans l'illustration 3.3. Vous utiliserez généralement l'élément fenêtre montré avec un tiroir ouvert sur la gauche. Il s'agit d'un objet de confort qui facilite la création et la connexion d'une fenêtre, de son tiroir et du contenu de celui-ci. Si vous glissez-déposez cette fenêtre sur la fenêtre Nib principale de votre projet, vous obtiendrez les trois instances visibles dans la rangée basse de

l'illustration 3.4. Vous les utiliserez comme ceci :

- Instance NSDrawer : Pour connecter des gestionnaires d'Events à l'objet drawer, sélectionnez l'instance NSDrawer et ouvrez la fenêtre Info dans le panneau AppleScript.
- Instance ParentWindow : Pour ajouter des éléments d'interface à la fenêtre qui possède le tiroir, double-cliquez sur l'instance ParentWindow pour ouvrir la fenêtre. Vous pouvez aussi utiliser la fenêtre Info pour connecter des gestionnaires d'Events à cette fenêtre.
- Instance DrawContentsView : Pour ajouter des éléments d'interface au tiroir, double-cliquez sur l'instance DrawContentsView pour ouvrir une fenêtre (comme dans l'illustration 3.5). Vous pouvez aussi utiliser la fenêtre Info pour connecter des gestionnaires d'Events à cette fenêtre. Vous constaterez que la fenêtre Info identifie cette instance comme étant "NSView (custom)".

Une fois que vous avez ajouté la fenêtre et le tiroir dans Interface Builder, vous devrez suivre ces étapes pour montrer cette fenêtre dans votre application, et autoriser l'utilisateur à ouvrir et fermer le tiroir. Vous pouvez montrer la fenêtre en connectant un gestionnaire launched (page 135) à l'objet "File's Owner" (lequel représente l'application) dans la fenêtre Nib. Voir la classe application (page 29) pour plus d'informations sur "File's Owner".

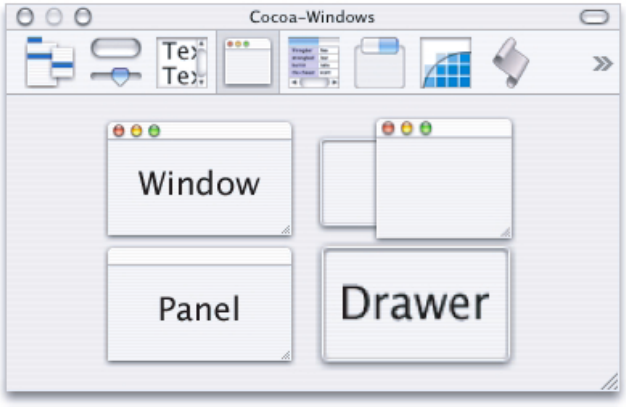

FIG. 3.3 - *Le panneau "Cocoa-Windows" d'Interface Builder, avec les tiroirs*

Si vous nommez applescriptement la fenêtre tiroir "main" (dans le panneau AppleScript de la fenêtre Info dans Interface Builder), votre gestionnaire Launched devrait ressembler au gestionnaire qui suit, extrait de l'application "Drawer" distribuée avec AppleScript Studio :

```
on launched theObject
  show window "main"
end launched
```

Pour autoriser l'utilisateur à ouvrir ou fermer le tiroir, vous devrez ajouter un bouton titré par exemple "Ouvrir le tiroir". Vous pourrez alors connecter un gestionnaire clicked (page 348) à ce bouton pour :

- ouvrir ou fermer le tiroir en accord avec l'état courant du tiroir
- régler le titre du bouton afin qu'il reflète l'état du tiroir (comme "Ouvrir le tiroir" lorsque le tiroir est fermé et "Fermer le tiroir" lorsque celui-ci est ouvert)

Vous pouvez utiliser la commande open drawer (page 240) pour ouvrir ou fermer le tiroir. Par exemple, l'instruction suivante ouvre le tiroir nommé "drawer" :

```
tell drawer "drawer" to open drawer
```

Il est aussi possible d'instancier un tiroir par lui-même (sans une fenêtre parent ou un "content view") en glissant-déposant l'objet étiqueté "Drawer" du panneau "Cocoa-Windows" sur la fenêtre Nib dans Interface Builder. Dans ce cas, vous obtiendrez juste l'instance NSDrawer et vous devrez le connecter vous-même à une fenêtre et à un content view.

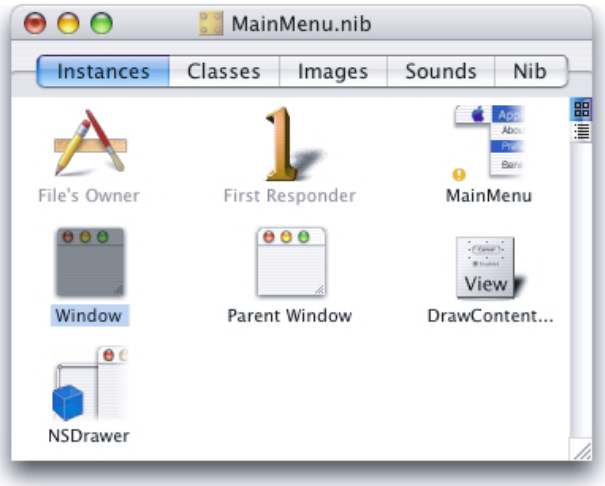

FIG. 3.4 - *La fenêtre MainMenu.nib après ajout d'une fenêtre tiroir*

Pour plus d'informations, voir Drawers dans la documentation Cocoa.

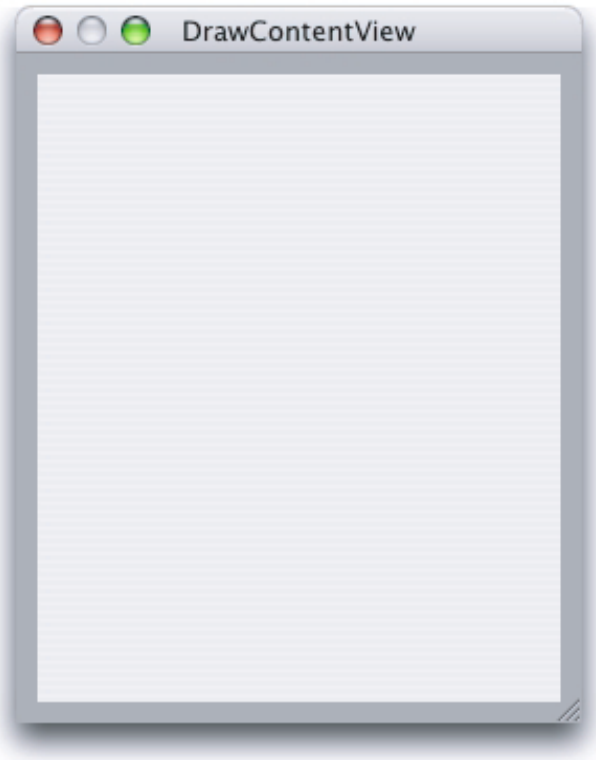

FIG. 3.5 - *Le content view d'un objet drawer (tiroir)*

Propriétés des objets de la classe Drawer

En plus des propriétés qu'il hérite de responder (page 68), un objet drawer possède ces propriétés :

content size

Accès : lecture / écriture
Classe : *point*
La taille du content view du tiroir ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; par exemple, {200, 100} indiquerait une largeur de 200 et une hauteur de 100 ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

content view

Accès : lecture / écriture
Classe : *view* (page 229)
Le content view du tiroir (décrit plus haut), lequel contient toutes ses

sous-views ; pour des informations de même nature, voir la propriété *content view* de la classe window (page 76)

edge

Accès : lecture uniquement
Classe : *une des constantes* de rectangle edge (page 186)
Le bord de la fenêtre sur lequel le tiroir est attaché

leading offset

Accès : lecture / écriture
Classe : *real*
Pour un tiroir qui s'ouvre sur le bord gauche ou droit, la distance entre le bord supérieur de la fenêtre et le bord supérieur du tiroir.
Pour un tiroir qui s'ouvre sur le bord supérieur ou inférieur, la distance entre le bord gauche de la fenêtre et le bord gauche du tiroir.
Si *leading offset* vaut 0, le bord (droit ou supérieur en fonction du côté choisi pour l'ouverture) du tiroir est au même niveau que le bord de la fenêtre ; si vous lancez l'application "Drawer" (visible dans l'illustration 3.2), vous pourrez ajuster "leading offset" pour voir le tiroir se déplacer en relation avec la fenêtre

maximum content size

Accès : lecture / écriture
Classe : *point*
La taille maximale du content view du tiroir ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical}, identique à la propriété *content size* plus haut

minimum content size

Accès : lecture / écriture
Classe : *point*
La taille minimale du content view du tiroir ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical}, identique à la propriété *maximum content size* plus haut

parent window

Accès : lecture / écriture
Classe : *window* (page 76)
La fenêtre associée au tiroir

preferred edge

Accès : lecture / écriture
Classe : *une des constantes* de rectangle edge (page 186)
Le bord (ou côté) préféré sur lequel s'ouvre le tiroir ; par défaut, cette propriété vaut `left edge` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder ; notez que cette propriété peut être ignorée s'il n'y a pas la place pour ouvrir le tiroir dans cette direction

state

Accès : lecture / écriture
Classe : *une des constantes* de drawer state (page 181)
L'état ouvert / fermé du tiroir

trailing offset

Accès : lecture uniquement
Classe : *real*
Pour un tiroir qui s'ouvre sur le bord gauche ou droit, la distance entre le bord inférieur de la fenêtre et le bord inférieur du tiroir.
Pour un tiroir qui s'ouvre sur le bord supérieur ou inférieur, la distance entre le bord droit de la fenêtre et le bord droit du tiroir.
Si *trailing offset* vaut 0, le bord (gauche ou inférieur en fonction du côté choisi pour l'ouverture) du tiroir est au même niveau que le bord de la fenêtre ; si vous lancez l'application "Drawer" (visible dans l'illustration 3.2), vous pourrez ajuster "trailing offset" pour voir le tiroir se déplacer en relation avec la fenêtre.

Éléments des objets de la classe Drawer

Un objet drawer peut contenir les éléments listés ci-dessous. Votre script peut spécifier la plupart des éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`box` (page 197)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets box du tiroir

`browser` (page 363)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets browser du tiroir

`button` (page 254)
Spécifier par :"Les formes-clés standards" (page 15)

Les objets button du tiroir

`clip view` (page 202)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets clip view du tiroir

`color well` (page 271)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets color well du tiroir

`combo box` (page 274)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets combo box du tiroir

`control` (page 280)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets control du tiroir

`image view` (page 285)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets image view du tiroir

`matrix` (page 289)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets matrix du tiroir

`movie view` (page 296)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets movie view du tiroir

`outline view` (page 392)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets outline view du tiroir

`popup button` (page 301)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets popup button du tiroir

`progress indicator` (page 306)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets progress indicator du tiroir

`scroll view` (page 213)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets scroll view du tiroir

`secure text field` (page 311)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets secure text field du tiroir

`slider` (page 315)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets slider du tiroir

`split view` (page 218)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets split view du tiroir

`stepper` (page 320)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets stepper du tiroir

`tab view` (page 221)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets tab view du tiroir

`table header view` (page 402)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets table header view du tiroir

`table view` (page 403)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets table view du tiroir

`text field` (page 325)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets text field du tiroir

`text view` (page 569)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets text view du tiroir

`view` (page 229)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets view du tiroir

Commandes supportées par les objets de la classe Drawer

Votre script peut envoyer les commandes suivantes à un objet drawer :

`close drawer` (page 239)

`open drawer` (page 240)

Events supportés par les objets de la classe Drawer

Un objet drawer supporte les gestionnaires répondant aux Events suivants :

Drawer

`closed` (page 129)
`opened` (page 143)
`should close` (page 151)
`should open` (page 152)
`will close` (page 161)
`will open` (page 165)
`will resize` (page 167)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

Exemples

Pour un aperçu du fonctionnement des tiroirs, voir la description plus haut de cette classe. Pour un exemple détaillé, voir l'application "Drawer" distribuée avec AppleScript Studio.

scroll view

Pluriel : `scroll views`
Hérite de : `view` (page 229)
Classe Cocoa : NSScrollView

Fournit la possibilité de faire défiler le contenu d'un document qui est trop grand pour être affiché dans son intégralité.

En plus de gérer le défilement, un objet scroll view peut afficher des règles et des ascenseurs verticaux et horizontaux (en fonction de la manière dont il est configuré). La plupart des défilements sont gérés automatiquement par le scroll view, mais voir la commande scroll (page 339) pour le mécanisme utilisé pour faire défiler le contenu d'un text view (page 569).

Un certain nombre de classes AppleScript Studio comporte automatiquement un scroll view, y compris les classes outline view (page 392), table view (page 403) et text view (page 569). Dans Interface Builder, vous pouvez incorporer n'importe quel objet d'interface dans un scroller en sélectionnant ces objets, en choisissant "Make subviews of" du menu "Layout", puis en choisissant "Scroll View". Vous pouvez utiliser ce même mécanisme pour transformer les views d'un objet box (page 197), split view (page 218) ou tab view (page 221) en sous-views.

Vous pouvez aussi utiliser ce mécanisme pour grouper des objets dans une view personnalisée (définie par vous) ou d'autres types de views (comme un objet matrix (page 289)). Pour faire cela, suivez ces étapes :

1. Sélectionnez les objets à grouper.
2. Choisissez "Make subviews of" du menu "Layout" et choisissez "Custom View".
3. Sélectionnez la view personnalisée obtenue. Par défaut, ce sera un objet view (page 229) (de la classe NSView).
4. Dans le panneau "Custom Class" de la fenêtre Info, sélectionnez le type de classe désiré.

Pour des informations de même nature, voir Drawing and Views dans la documentation Cocoa.

Propriétés des objets de la classe Scroll View

En plus des propriétés qu'il hérite de view (page 229), un objet scroll view possède ces propriétés :

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du fond ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur bleue peut être représentée par {0, 0, 65535} ; par défaut la couleur est réglée sur la couleur blanche {65535, 65535, 65535}

border type

Accès : lecture / écriture
Classe : *une des constantes* de border type (page 175)
Le type de contour du scroll view

content size

Accès : lecture uniquement
Classe : *point*
La taille de la view du scroll view ; la taille est exprimée sous forme d'une liste de deux nombres entiers {horizontal, vertical} ; par exemple, {200, 100} indiquerait une largeur de 200 et une hauteur de 100 ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

content view

Accès : lecture / écriture
Classe : *view* (page 229)
Le content view ; l'objet clip view (page 202) qui découpe l'affichage du document ; pour des informations de même nature, voir la propriété *content view* de la classe window (page 76)

document view

Accès : lecture / écriture
Classe : *view* (page 229)
Le document view que l'ascenseur fait défiler

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet scroll view dessine son fond ? Par défaut, cette propriété vaut `true`

dynamically scrolls

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet view défile dynamiquement ? Par défaut, cette propriété vaut `true`

has horizontal ruler

Accès : lecture / écriture
Classe : *boolean*
Le scroll view a-t-il une règle horizontale ?

has horizontal scroller

Accès : lecture / écriture
Classe : *boolean*
Le scroll view a-t-il un ascenseur horizontal ?

has vertical ruler

Accès : lecture / écriture
Classe : *boolean*
Le scroll view a-t-il une règle verticale ?

has vertical scroller

Accès : lecture / écriture
Classe : *boolean*
Le scroll view a-t-il un ascenseur vertical ?

horizontal line scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de lignes horizontales à faire défiler

horizontal page scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de pages horizontales à faire défiler

horizontal ruler view

Accès : lecture / écriture
Classe : *n'importe*
La règle horizontale

horizontal scroller

Accès : lecture / écriture
Classe : *n'importe*
L'ascenseur horizontal

line scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de lignes à faire défiler

page scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de pages à faire défiler

rulers visible

Accès : lecture / écriture
Classe : *boolean*
Les règles sont-elles visibles ?

vertical line scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de lignes verticales à faire défiler

vertical page scroll

Accès : lecture / écriture
Classe : *real*
Le nombre total de pages verticales à faire défiler

vertical ruler view

Accès : lecture / écriture
Classe : *n'importe*
La règle verticale

vertical scroller

Accès : lecture / écriture
Classe : *n'importe*
L'ascenseur vertical

visible document rect

Accès : lecture uniquement

Classe : *bounding rectangle*
Les limites visibles du document

Éléments des objets de la classe Scroll View

Un objet scroll view peut uniquement contenir les éléments qu'il hérite de la classe view (page 229).

Events supportés par les objets de la classe Scroll View

Un objet scroll view supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Comme beaucoup d'objets "résident" automatiquement sur un scroll view, y compris les objets outline view (page 392), table view (page 403) et text view (page 569), les applications AppleScript Studio ont souvent besoin d'inclure un scroll view dans la spécification d'un objet. La ligne suivante est extraite de l'application "Table" distribuée avec AppleScript Studio.

```
tell table view "contacts" of scroll view "contacts"
  of window of theObject to update
```

Vous pouvez accéder aux propriétés d'un scroll view avec des instructions comme celle qui suit, laquelle règle une variable suivant si le scroll view a ou non un ascenseur vertical :

```
set hasVertScroller to has vertical scroller of scroll view "contacts" of
  window of theObject
```

Version

Le support des Events de glisser-déposer est apparu avec la version 1.2 d'AppleScript Studio.

split view

Pluriel : `split views`
Hérite de : `view` (page 229)
Classe Cocoa : NSSplitView

Empile plusieurs sous-views dans une seule view pour coordonner les modifications de leurs tailles relatives.

Les barres de partage entre les views peuvent être horizontales ou verticales suivant si elles sont arrangées verticalement ou côte à côte. Dans l'illustration 3.6, un objet outline view (page 392) apparaît au-dessus d'un objet table view (page 403).

Voir la classe scroll view (page 213) pour plus d'informations sur la manière de mettre dans Interface Builder des objets dans les sous-views d'un split view ou d'une autre view.

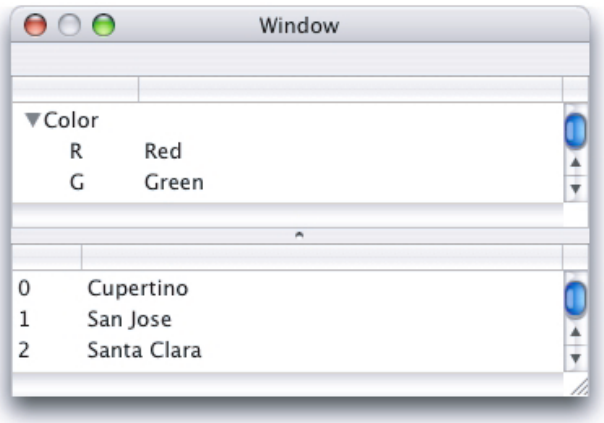

FIG. 3.6 - *Un split view contenant un outline view et un table view*

Propriétés des objets de la classe Split View

En plus des propriétés qu'il hérite de view (page 229), un objet split view possède ces propriétés :

pane splitter

Accès : lecture / écriture
Classe : *boolean*
Y a-t-il un "pane splitter" ? (voir la barre horizontale au milieu de l'illustration 3.6)

vertical

Accès : lecture / écriture
Classe : *boolean*
Le "splitter" est-il vertical ?

Éléments des objets de la classe Split View

Un objet split view peut uniquement contenir les éléments qu'il hérite de view (page 229).

Events supportés par les objets de la classe Split View

Un objet split view supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

Split View

`resized sub views` (page 244)
`will resize sub views` (page 247)

View

`bounds changed` (page 243)

Exemples

Le script suivant obtient la propriété *vertical* d'un split view dans l'application "Mail Search" distribuée avec AppleScript Studio. Ce script ne fait pas partie de l'application, mais vous pouvez le lancer depuis l'application Éditeur de Scripts pour accéder à cette propriété de l'application "Mail Search". Des instructions similaires fonctionneront à l'intérieur d'un script d'une application AppleScript Studio (bien que n'aurez pas besoin de l'encadrer par un bloc `tell`).

```
tell application "Mail Search"
  tell front window
    set isVertical to vertical of first split view
    -- Do something based on result
  end tell
end tell
```

Vous pourriez aussi avoir besoin de vous référer à un split view pour accéder à une autre view de votre application. L'instruction suivante extraite de l'application "Mail Search" spécifie un scroll view sur un split view d'une fenêtre :

```
tell scroll view "mailboxes" of split view 1 of theWindow
  -- Access properties or subviews of the scroll view.
end tell
```

Version

Le support des Events de glisser-déposer est apparu avec la version 1.2 d'AppleScript Studio.

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" se nommait "Watson".

tab view

Pluriel : `tab views`
Hérite de : `view` (page 229)
Classe Cocoa : NSTabView

Fournit une manière facile de présenter des informations sur plusieurs pages.

La view contient une rangée d'onglets qui donne l'apparence d'un dossier d'onglets, comme dans l'illustration 3.7. L'utilisateur sélectionne la page désirée en cliquant sur l'onglet approprié ou en utilisant les flèches du clavier pour se déplacer entre les pages. Chaque page affiche une hiérarchie de views fournie par votre application. Chaque onglet, et sa hiérarchie de views associée, est représenté par un objet tab view item (page 227).

Voir la classe scroll view (page 213) pour plus d'informations sur la manière de mettre des objets dans des sous-views d'un tab view ou d'une autre view dans Interface Builder.

FIG. 3.7 - *Un tab view avec trois onglets*

Propriétés des objets de la classe Tab View

En plus des propriétés qu'il hérite de view (page 229), un objet tab view possède ces propriétés :

content rect

Accès : lecture uniquement
Classe : *bounding rectangle*
Les limites du contenu du tab view ; une liste de quatre nombres {gauche, bas, droite, haut} ; le rectangle est mis au point par le système de coordonnées du tab view ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

control size

Accès : lecture / écriture
Classe : *une des constantes* de control size (page 180)
La taille des onglets ; par défaut cette propriété vaut `regular size` ; vous pouvez la régler dans Interface Builder (avec la case à cocher “Small Tabs” du panneau “Attributes” de la fenêtre Info)

control tint

Accès : lecture / écriture
Classe : *une des constantes* de control tint (page 180)
La teinte des onglets ; par défaut elle vaut `default tint`

current tab view item

Accès : lecture / écriture
Classe : *tab view item* (page 227)
Le tab view item courant

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l’objet tab view dessine son fond ? Par défaut, cette propriété vaut `true` ; voir la description plus haut pour plus d’informations pour savoir à quel moment vous pouvez régler cette propriété dans la fenêtre Info d’Interface Builder

tab type

Accès : lecture / écriture
Classe : *une des constantes* de tab view type (page 188)
Le type d’onglet (comme un onglet en haut d’un tab view (page 221)) ; vous pouvez régler la direction des onglets (haut, bas, gauche ou droite) dans Interface Builder, bien que les directions bas, gauche et droite ne fonctionnent que depuis la version de Cocoa livrée avec Mac OS X version 10.2

truncated labels

Accès : lecture / écriture
Classe : *boolean*
Faut-il si nécessaire tronquer les étiquettes ? Vous pouvez régler cette propriété dans la fenêtre Info d’Interface Builder

Éléments des objets de la classe Tab View

En plus des éléments qu'il hérite de la classe view (page 229), un objet tab view peut contenir les éléments listés ci-dessous. Votre script peut spécifier la plupart des éléments à l'aide des références décrites dans "Les formes-clés standards" (page 15).

`tab view item` (page 227)
Spécifier par :"Les formes-clés standards" (page 15)
Les tab view item de la view, un par onglet dans la view

Events supportés par les objets de la classe Tab View

Un objet tab view supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

awake from nib (page 123)

Tab View

selected tab view item (page 245)

should select tab view item (page 246)

will select tab view item (page 248)

View

bounds changed (page 243)

Exemples

Étant donné les noms appropriés aux objets, vous pouvez utiliser la terminologie suivante pour régler le texte d'un text field (page 325) dans un tab view :

```
set contents of text field "textFieldName" of view of tab view item
"tabViewItemName" of tab view "tabViewName" of window "windowname"
```

Les éléments d'un tab view item (page 227) sont généralement sur une view (page 229) qui est elle-même sur un tab view item (page 227), lequel représente la view dans la phrase `of view of tab view item` de cet exemple. Toutefois, à partir de la version 1.2 d'AppleScript Studio, vous n'aurez plus besoin de spécifier la view, et vous pourrez utiliser l'instruction suivante, forme simplifiée de la précédente :

```
set contents of text field "textFieldName" of tab view item "tabViewItemName"
of tab view "tabViewName" of window "windowname"
```

Vous pouvez utiliser un tab view pour simuler le "zapping" du contenu d'une view. Dans Interface Builder, glissez-déposez un objet tab view du panneau "Cocoa-Containers" sur la fenêtre visée. Avec le tab view sélectionné, cochez le bouton "Tabless". Vous pouvez alors choisir le style, entre avoir une plateforme ombrée ou avoir aucun cadre visible. Si vous choisissez aucun cadre visible, vous pourrez aussi choisir si le tab view pourra dessiner son fond.

Vous pourrez encore mettre les objets d'interface voulus sur chaque tab view item (page 227). Comme il n'y a aucun onglet sélectionnable, la fenêtre Info fournit un compteur permettant de passer d'un onglet à l'autre. Pour

arriver à afficher ce compteur, il faudra que les objets tab view item soient sélectionnés, pour se faire, vous devrez double-cliquer sur le tab view pour sélectionner ses tab view items, après il vous suffira de cliquer sur le compteur (visible dans l'illustration 3.8) dans le panneau "Attributes" pour afficher le contenu de chaque tab item view. Comme l'utilisateur devra pouvoir sélectionner un onglet parmi ceux présentés (but recherché ici), vous devrez modifier le tab view item généralement affiché de façon programmé, avec des instructions comme celle qui suit où il est supposé que vous avez baptisé le premier tab view item "tabViewItem1" :

```
tell tab view "tabview" of window "main"
  set the current tab view item to tab view item "tabViewItem1"
end
```

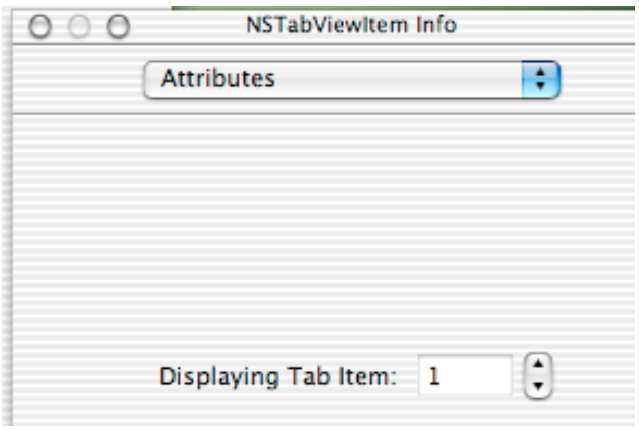

FIG. 3.8 - *Le compteur permettant de "switcher" d'un onglet à l'autre*

Version

Le support des Events de glisser-déposer est apparu avec la version 1.2 d'AppleScript Studio.

Les tab view sont implémentés dans AppleScript Studio par la classe Cocoa NSTabView. Avant la version 10.2 de Mac OS X, la classe NSTabView supportait uniquement les onglets en haut.

Depuis la version 1.2 d'AppleScript Studio, un script peut dire `button 1 of tab view item 1 of tab view 1` au lieu de `button 1 of view of tab view item 1 of tab view 1`, bien que la version longue fonctionne toujours (et se lancera avec toutes les versions d'AppleScript Studio). Voir la section "Exemples" plus haut pour un autre exemple.

tab view item

Pluriel : `tab view items`
Hérite de : `personne`
Classe Cocoa : NSTabViewItem

Représente un onglet dans un tab view (page 221).

Lorsqu'un utilisateur clique sur un onglet, le tab view affiche la page fournie par l'application. Un tab view garde un tab view item basé sur un tableau, un pour chaque onglet de la view. Le tab view de l'illustration 3.7 comporte trois tab view item.

Lorsque vous glissez un tab view (page 221) depuis le panneau "Cocoa-Containers" d'Interface Builder, il contient par défaut deux tab view item. Vous pouvez régler le nombre de tab view item dans la fenêtre Info d'Interface Builder.

Propriétés des objets de la classe Tab View Item

Un objet tab view item possède ces propriétés :

color

Accès : lecture / écriture
Classe : *RGB color*
La couleur RVB du tab view item ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur bleue peut être représentée par {0, 0, 65535} ; par défaut elle est réglée sur la couleur blanche {65535, 65535, 65535}

label

Accès : lecture / écriture
Classe : *Unicode text*
L'étiquette du tab view item ; vous pouvez régler l'étiquette dans Interface Builder

tab state

Accès : lecture uniquement
Classe : *une des constantes* de tab state (page 188)
L'état du tab view item

tab view

Accès : lecture uniquement

Classe : *tab view* (page 221)
Le tab view qui contient ce tab view item

view

Accès : lecture / écriture
Classe : *view* (page 229)
La view du tab view item (sur lequel vous placez les objets d'interface du tab view item)

Events supportés par les objets de la classe Tab View Item

Un objet tab view item supporte les gestionnaires répondant aux Events suivants :

Nib

`awake from nib` (page 123)

Exemples

Le gestionnaire suivant est extrait de l'application "Assistant" distribuée avec AppleScript Studio (depuis la version 1.1). Ce gestionnaire est appelé lorsque les propriétés du script ont besoin d'être mises à jour depuis le contenu des objets d'interface associés au tab view item. Comme le montre la première ligne de l'instruction `tell`, la terminologie pour accéder à un tab view (page 221) peut devenir très complexe. L'erreur courante avant la version 1.2 était d'omettre `view of` au début de l'instruction. Depuis la version 1.2, `view of` est optionnel.

À l'intérieur du bloc tell, les instructions pour recueillir l'information de chaque champ texte sont plus simples. Chaque instruction assigne une valeur à une propriété du script.

```
on updateValues(theWindow)
  tell view of tab view item infoPanelName of tab view "info panels"
      of box "border" of theWindow
    set my company to contents of text field "company"
    set my name to contents of text field "name"
    set my address to contents of text field "address"
    set my city to contents of text field "city"
    set my state to contents of text field "state"
    set my zip to contents of text field "zip"
```

```
    set my email to contents of text field "email"
  end tell
end updateValues
```

La section “Exemples” de la classe tab view (page 221) décrit comment utiliser un tab view pour simuler le “zapping” du contenu d’une view en passant d’un tab view item à l’autre.

view

Pluriel : views
Hérite de : responder (page 68)
Classe Cocoa : NSView

Une classe-résumé définissant l’architecture de base des dessins, de la gestion des Events et de l’impression d’une application.

Vos scripts en général n’interagissent pas directement avec les objets view ; ils interagissent plutôt avec la plupart des classes d’interface héritant de la classe view.

Vous pouvez créer et accéder à un objet view dans Interface Builder en suivant ces étapes :

- Glissez une instance de “CustomView” du panneau “Cocoa-Containers” sur la fenêtre visée. Cette view est par défaut un objet view de la classe NSView.

Pour modifier la classe en classe view personnalisée (définie par vous) ou en un autre type de classe, suivez ces étapes :

1. Sélectionnez la view personnalisée.
2. Dans le panneau “Custom Class” de la fenêtre Info, sélectionnez le nouveau type de classe.

Propriétés des objets de la classe View

En plus des propriétés qu’il hérite de responder (page 68), un objet view possède ces propriétés :

auto resizes

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l’objet view s’auto-redimensionne ?

bounds

Accès : lecture / écriture
Classe : *bounding rectangle*
La position et la taille de la view (à l'intérieur de sa super-view) ; les limites sont exprimées sous forme d'une liste de quatre nombres {gauche, bas, droite, haut} ; le rectangle limite est réglé à partir de {0, 0} dans le super-view ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

bounds rotation

Accès : lecture / écriture
Classe : *real*
La rotation des limites, en degrés ; par défaut elle vaut 0.0 ; des valeurs positives indiquent une rotation dans le sens des aiguilles d'une montre, négatives le sens inverse ; la rotation est exécutée avec comme origine l'origine du système des coordonnées, (0.0, 0.0), lequel n'a pas besoin de coïncider avec celui du cadre ou des limites du rectangle ; modifier cette valeur ne réaffiche pas la view ou la marque pas comme ayant besoin d'être affichée ; vous pouvez faire cela en réglant la propriété *needs display* sur `true`, ou en utilisant la commande update (page 119)

can draw

Accès : lecture uniquement
Classe : *boolean*
La view peut-elle être dessinée ? `true` si les commandes de dessin peuvent produire n'importe quel résultat, `false` dans l'autre cas ; cette propriété est utilisée lorsqu'est invoquée directement le dessinement, ainsi que les commandes lock focus (page 240) et unlock focus (page 241) ; toutefois ces commandes ne sont pas supportées dans la version 1.3 d'AppleScript Studio ; de plus, AppleScript Studio ne fournit pas de contrôle fin sur le dessinement, et la plupart des applications n'auront pas besoin de l'invoquer directement ; si votre application est une exception, voir NSView, ainsi que la documentation Cocoa sur cette classe

enclosing scroll view

Accès : lecture / écriture
Classe : *scroll view* (page 213)
La scroll view de la view (s'il y en a une) ; voir la description de la classe scroll view pour savoir comment incorporer dans Interface Builder une

view dans un scroll view

flipped

Accès : lecture uniquement
Classe : *boolean*
Est-ce que le système des coordonnées de la view est tourné ? Par défaut l'origine du système de coordonnées de la view est situé dans le coin inférieur gauche ; toutefois, pour certaines views, la valeur par défaut de cette propriété vaut `true`, signifiant que l'origine est dans le coin supérieur gauche ; cette propriété est en lecture uniquement, et il est peu probable que votre application soit concernée par cela ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées, lequel diffère de celui qui est utilisé par l'application Finder de Mac OS

needs display

Accès : lecture / écriture
Classe : *boolean*
La view a-t-elle besoin d'être affichée ? Régler cette propriété sur `true` provoquera le redessinement de la view jusqu'à une prochaine opportunité ; pour provoquer un redessinement immédiat, utiliser la commande update (page 119) ; noter que dans la version 1.3 d'AppleScript Studio, la propriété *needs display* n'est pas supportée par la classe window (page 76), mais l'est par la classe view

opaque

Accès : lecture uniquement
Classe : *boolean*
La view est-elle opaque ? Voir la description de la propriété *opaque* de la classe window (page 76)

position

Accès : lecture / écriture
Classe : *point*
La position de la view à l'intérieur de sa super-view sous forme d'une liste de deux nombres {gauche, bas} ; chaque view a son propre système de coordonnées, avec l'origine dans le coin supérieur gauche ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

size

Accès : lecture / écriture
Classe : *point*
La taille de la view ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; par exemple, {200, 100} indiquerait une largeur de 200 et une hauteur de 100 ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

super view

Accès : lecture / écriture
Classe : *view* (page 229)
La view contenant cette view

tag

Accès : lecture / écriture
Classe : *integer*
L'étiquette de la view ; vous pouvez régler l'étiquette pour certaines views, comme des views text field (page 325), dans la fenêtre Info d'Interface Builder

tool tip

Accès : lecture / écriture
Classe : *Unicode text*
La bulle d'aide de la view (texte devant être affiché si l'utilisateur laisse le curseur de la souris stationné quelques instants au-dessus de la view)

visible

Accès : lecture / écriture
Classe : *boolean*
La view est-elle visible ?

visible rect

Accès : lecture uniquement
Classe : *bounding rectangle*
La surface visible de la view ; une liste de quatre nombres, {gauche, bas, droite, haut} ; la view a son propre système de coordonnées ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

window

Accès : lecture / écriture
Classe : *window* (page 76)
La fenêtre contenant cette view

Éléments des objets de la classe View

Un objet view peut contenir les éléments listés ci-dessous. Vos scripts peuvent spécifier la plupart de ces éléments à l'aide des formes-clés décrites dans "Les formes-clés standards" (page 15).

`box` (page 197)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets box de la view

`browser` (page 363)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets browser de la view

`button` (page 254)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets button de la view

`clip view` (page 202)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets clip view de la view

`color well` (page 271)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets color well de la view

`combo box` (page 274)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets combo box de la view

`control` (page 280)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets control de la view

`image view` (page 285)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets image view de la view

`matrix` (page 289)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets matrix de la view

`movie view` (page 296)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets movie view de la view

`outline view` (page 392)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets outline view de la view

`popup button` (page 301)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets popup button de la view

`progress indicator` (page 306)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets progress indicator de la view

`scroll view` (page 213)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets scroll view de la view

`secure text field` (page 311)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets secure text field de la view

`slider` (page 315)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets slider de la view

`split view` (page 218)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets split view de la view

`stepper` (page 320)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets stepper de la view

`tab view` (page 221)
Spécifier par :"Les formes-clés standards" (page 15)
Les objets tab view de la view

`table header view` (page 402)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets table header view de la view

`table view` (page 403)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets table view de la view

`text field` (page 325)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets text field de la view

`text view` (page 569)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets text view de la view

`view` (page 229)
Spécifier par : "Les formes-clés standards" (page 15)
Les objets view de la view

Commandes supportées par les objets de la classe View

Votre script peut envoyer les commandes suivantes à un objet view :

`lock focus` (page 240) (non supportée dans la version 1.3 d'AppleScript Studio)
`register` (page 116)
`unlock focus` (page 241) (non supportée dans la version 1.3 d'AppleScript Studio)

Events supportés par les objets de la classe View

Un objet view supporte les gestionnaires répondant aux Events suivants. Pour connecter dans Interface Builder un gestionnaire d'Events à un objet view, mettez la fenêtre Nib de l'objet window (page 76) contenant la view en mode "outline" en cliquant sur la petite icone "outline" au-dessus de l'ascenseur vertical droit ; utilisez les triangles pour ouvrir l'objet window et les autres objets jusqu'à ce que l'objet view soit visible ; sélectionnez-le, puis connectez le gestionnaire d'Events dans le panneau "AppleScript" de la fenêtre Info.

Glisser-Déposer

`conclude drop` (page 481)

`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

La classe View est une classe-résumé que vous ne viserez généralement pas dans vos scripts, mais plutôt ses sous-classes, comme box (page 197), scroll view (page 213) ou tab view (page 221). La classe control (page 280) hérite aussi de la classe View, aussi toutes les sous-classes de control héritent des propriétés et des éléments de la classe View (bien que certains éléments de View, comme un movie view (page 296) ou un progress indicator (page 306), ne soient pas très utiles, par exemple, à un objet control comme un objet button (page 254)).

Vous pouvez utiliser le script suivant dans l'application Éditeur de Scripts pour faire pivoter de 50 degrés le texte dans un text view (page 569). Des instructions similaires fonctionneront à l'intérieur d'une application AppleScript Studio (bien que vous n'aurez pas besoin de l'instruction `tell application`).

```
tell application "rotate"
  tell window 1
    tell scroll view 1
      tell text view 1
        set bounds rotation to 50.0
        set needs display to true
      end tell
    end tell
  end tell
end tell
```

Version

La propriété *needs display* n'est pas supportée par la classe window (page 76) dans la version 1.3 d'AppleScript Studio, mais elle l'est par la classe View.

Les commandes lock focus (page 240) et unlock focus (page 241) ne sont pas supportées par la version 1.3 d'AppleScript Studio.

Chapitre 2

Commandes

Les objets basés sur les classes de la suite Container View supporte les commandes suivantes. Une **commande** est un mot ou une phrase qu'un script peut utiliser pour demander une action. Pour déterminer les commandes supportées par chaque classe, voir les descriptions propres à chaque classe.

close drawer 239
lock focus 240
open drawer 240
unlock focus 241

`close drawer`

Ferme le drawer spécifié.

Syntaxe

`close drawer` *reference* obligatoire

Paramètres

reference

La référence de l'objet drawer (page 204) à fermer

Exemples

Avec une fenêtre portant le nom AppleScript “main” contenant un tiroir nommé “drawer”, vous pouvez fermer ce tiroir avec une instruction `tell` comme celle qui suit :

```
tell window "main"
  tell drawer "drawer" to close
end tell
```

lock focus

Non supportée par la version 1.3 d'AppleScript Studio. Verrouille la mise au point d'une view pour la préparer à son redessinement.

Syntaxe

`lock focus`	*reference*	obligatoire

Paramètres

reference
: La référence de l'objet view (page 229) pour lequel la mise au point va être verrouillée

open drawer

Ouvre le tiroir spécifié.

Syntaxe

`open drawer`	*reference*	obligatoire
`[on]`	*une constante*	facultatif

Paramètres

reference
: La référence de l'objet drawer (page 204) à ouvrir

`[on]` *une des constantes* de rectangle edge (page 186)
: Le côté de la fenêtre au niveau duquel devra s'ouvrir le tiroir

Exemples

Avec une fenêtre portant le nom AppleScript “main” contenant un objet drawer (page 204) nommé “drawer”, vous pouvez ouvrir ce tiroir sur le côté gauche de la fenêtre avec une instruction `tell` comme celle qui suit :

```
tell window "main"
  tell drawer "drawer" to open drawer on left edge
end tell
```

unlock focus

Non supportée par la version 1.3 d’AppleScript Studio.

Syntaxe

`unlock focus` *reference* obligatoire

Paramètres

reference

La référence de l’objet view (page 229) pour lequel la mise au point va être déverrouillée

Chapitre 3

Events

Les objets basés sur les classes de la suite Container View supportent les gestionnaires répondant aux Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

.

bounds changed 243
resized sub views 244
selected tab view item 245
should select tab view item 246
will resize sub views 247
will select tab view item 248

`bounds changed`

Appelé après que les limites d'un objet view (page 229) aient été modifiées.

Syntaxe

`bounds changed` *reference* obligatoire

Paramètres

reference

La référence de l'objet view (page 229) dont les limites ont changé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Bounds Changed à un objet view (page 229), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour réagir à toute modification des limites. Pour mesurer cette modification, vous devrez enregistrer les anciennes limites de la view et les comparer avec les limites courantes.

```
on bounds changed theObject
  (* Perform operations here after bounds changed. *)
end bounds changed
```

resized sub views

Appelé après que les sous-views d'un objet view (page 229) soient redimensionnées.

Syntaxe

`resized sub views` *reference* obligatoire

Paramètres

reference

La référence de l'objet view (page 229) dont les sous-views sont redimensionnées

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Resized Sub Views à un objet view (page 229), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Pour déterminer la modification de la taille de la view, vous devrez enregistrer l'ancienne taille de la view (ce que vous pouvez faire avec un gestionnaire will resize sub views (page 247)) et la comparer avec la taille

courante. Vous pouvez utiliser ce gestionnaire pour exécuter toute opération nécessaire une fois que les sous-views sont redimensionnées.

```
on resized sub views theObject
  (* Perform operations here after sub views resized. *)
end resized sub views
```

selected tab view item

Appelé après qu'un tab view item (page 227) soit sélectionné, indiquant que le tab view item courant a été modifié.

Un tab view item représente un onglet dans un tab view (page 221).

Syntaxe

`selected tab view item`	*tab view*	obligatoire
`[tab view item]`	*tab view item*	facultatif

Paramètres

tab view (page 221)
le tab view dont le tab view item a été sélectionné

`[tab view item]` *tab view item* (page 227)
le tab view item qui a été sélectionné

Exemples

Le gestionnaire Selected Tab View Item suivant est extrait de l'application "Assistant" distribuée avec AppleScript Studio depuis la version 1.1. Ce gestionnaire est appelé lorsque le tab view item (page 227) courant a été remplacé. C'est un bon endroit pour exécuter toute opération avant l'affichage du contenu de l'onglet.

```
on selected tab view item theObject tab view item tabViewItem
  -- We will give the new info panel a chance to
  -- prepare it's data values
  prepareValues(window of theObject)
    of infoPanelWithName(name of tabViewItem)
end selected tab view item
```

should select tab view item

Appelé pour déterminer si les objets tab view item (page 227) des objets devront être sélectionnés, très probable car l'utilisateur a cliqué sur l'onglet associé.

Ce gestionnaire peut retourner `false` pour refuser la sélection de l'élément (aussi il n'y aura pas de changement de tab item) ou `true` pour l'autoriser. Un tab view item (page 227) représente un onglet dans un tab view (page 221).

Syntaxe

`should select tab view item`	*tab view*	obligatoire
`[tab view item]`	*tab view item*	facultatif

Paramètres

tab view (page 221)
: le tab view dont le tab view item pourrait être sélectionné

`[tab view item]` *tab view item* (page 227)
: le tab view item

Résultats

boolean
: Retourne `true` pour autoriser la sélection ; `false` pour l'interdire. Si vous implémentez ce gestionnaire, vous devrez toujours retourner une valeur booléenne

Exemples

L'exemple suivant de gestionnaire Should Select Tab View Item appelle le gestionnaire `shouldSelectTabViewItem`, écrit par vous, pour déterminer s'il doit autoriser la sélection de l'élément, puis retourne la valeur appropriée. Vous pourriez aussi à la place exécuter une validation dans le gestionnaire lui-même ou vérifier certaines propriétés.

```
on should select tab view item theObject
  --Check property, perform test, or call handler to see if OK
  -- to select tab view item specified by theObject
  set allowSelection to shouldSelectTabViewItem(theObject)
```

```
  return allowSelection
end should select tab view item
```

L'application "Assistant", disponible depuis la version 1.1 d'AppleScript Studio, inclut un gestionnaire Should Select Tab View Item qui examine chaque panel avant de décider s'il doit autoriser une modification dans l'onglet sélectionné.

will resize sub views

Appelé lorsque les sous-views d'un objet view (page 229) sont sur le point d'être redimensionnées.

Le gestionnaire ne peut pas annuler le redimensionnement, mais peut le préparer.

Syntaxe

`will resize sub views` *reference* obligatoire

Paramètres

reference

La référence de l'objet view (page 229) dont les sous-views sont sur le point d'être redimensionnées

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Resize Sub Views à un objet view (page 229), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer les sous-views à être redimensionnées. Par exemple, vous pourriez enregistrer la taille courante des sous-views.

```
on will resize sub views theObject
  (* Perform any operations to prepare for resizing of subviews.*)
end will resize sub views
```

will select tab view item

Appelé lorsqu'un tab view item (page 227) est sur le point d'être sélectionné. Un tab view item représente un onglet dans un tab view (page 221). Ce gestionnaire ne peut pas annuler la sélection, mais peut la préparer.

Syntaxe

`will select tab view item`	*tab view*	obligatoire
`[tab view item]`	*tab view item*	facultatif

Paramètres

tab view (page 221)
: le tab view dont le tab view item est sur le point d'être sélectionné

`[tab view item]` *tab view item* (page 227)
: le tab view item qui est sur le point d'être sélectionné

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Select Tab View Item à un objet view (page 229), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le tab view item à être sélectionné.

```
on will select tab view item theObject
  (* Perform operations to prepare for selection of the item. *)
end will select tab view item
```

Quatrième partie

Control View Suite

Cette partie décrit la terminologie de la suite Control View d'AppleScript Studio.

La suite Control View définit un certain nombre de classes permettant d'implémenter et de travailler avec des objets control (page 280). Les **controls** sont des objets graphiques qui provoquent des actions immédiates ou des résultats visibles lorsqu'un utilisateur les manipulent avec la souris.

La plupart des classes de cette suite héritent de la classe view (page 229), soit directement, soit par l'intermédiaire de la classe control (page 280). La suite Control View définit aussi plusieurs Events fonctionnant avec les actions de l'utilisateur invoquant des contrôles.

Les classes, commandes et Events de la suite Control View sont décrits dans les chapitres suivants :

Classes 253
Commandes 333
Events 345

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Control View contient les classes suivantes :

action cell 254
button 254
button cell 261
cell 264
color well 271
combo box 274
combo box item 279
control 280
image cell 284
image view 285
matrix 289
movie view 296
popup button 301
progress indicator 306
secure text field 311
secure text field cell 314
slider 315
stepper 320
text field 325
text field cell 330

action cell

Pluriel : `action cells`
Hérite de : `cell` (page 264)
Classe Cocoa : NSActionCell

Définit la surface active d'un objet control (page 280) ou d'une de ses sous-classes.

Comme la surface active d'un control, un objet action cell fait trois choses : il exécute genéralement l'affichage du texte ou de la vignette, il fournit le contrôle avec une cible ou une action et il gère les déplacements de la souris (curseur) en illuminant (dans le sens des détections radars) correctement son aire et en envoyant les messages action basés sur les mouvements du curseur à ses cibles.

Events supportés par les objets de la classe Action Cell

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires d'Events.

Exemples

Voir les exemples de cell (page 264).

button

Pluriel : `buttons`
Hérite de : `control` (page 280)
Classe Cocoa : NSButton

Sous-classe de control qui intercepte les Events "mouse-down" (clic de souris) et qui initie l'appel du gestionnaire lorsqu'il est cliqué ou appuyé.

Un objet button contient un seul objet button cell (page 261). L'illustration 4.1 montre un bouton.

Vous trouverez tout un assortiment de boutons dans le panneau "Cocoa-Controls" d'Interface Builder. Vous pouvez régler la plupart des attributs d'un objet button dans la fenêtre Info d'Interface Builder. Vous pouvez aussi modifier les types de bouton dans vos scripts, en utilisant les constantes définies dans button type (page 177).

FIG. 4.1 - *Un bouton*

Propriétés des objets de la classe Button

En plus des propriétés qu'il hérite de la classe control (page 280), un objet button possède ces propriétés :

allows mixed state

Accès : lecture / écriture
Classe : *boolean*
Le bouton autorise-t-il l'état mixte ? (voir aussi la propriété *state*) ; certains boutons peuvent uniquement indiquer deux états, comme actif ou inactif ; un état mixte indique plus que deux états ; supposons qu'une "checkbox" sert à vérifier du texte , si tout le texte sélectionné est en style gras, la "checkbox" est active, si aucun mot du texte n'est en gras, elle est inactive, par contre, si quelques mots mais pas tous sont en gras, son état est mixte

alternate image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image du bouton lorsqu'il est dans un état alterné ; voir la section "Discussion"

alternate title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre du bouton lorsqu'il est dans un état alterné ; voir la section "Discussion"

bezel style

Accès : lecture / écriture

Classe : *une des constantes* de bezel style (page 175)
L'apparence du contour d'un bouton

bordered

Accès : lecture / écriture
Classe : *boolean*
Le bouton a-t-il une bordure ?

button type

Accès : lecture / écriture
Classe : *une des constantes* de button type (page 177)
Le type de bouton

image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image du bouton ; voir la section "Discussion"

image position

Accès : lecture / écriture
Classe : *une des constantes* de cell image position (page 177)
La position de l'image dans le bouton, sous forme d'une liste de deux nombres {gauche, bas} ; chaque objet window (page 76) ou view (page 229) a son propre système de coordonnées, avec l'origine dans le coin inférieur gauche ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

key equivalent

Accès : lecture / écriture
Classe : *Unicode text*
Le raccourci clavier équivalent au clic sur le bouton ; par exemple, le raccourci clavier pourrait être la lettre "U" ou la combinaison Cmd + U ; vous pouvez régler cette propriété dans le panneau "Attributes" de la fenêtre Info d'Interface Builder, où, par exemple, vous pouvez régler un bouton pour être le bouton par défaut en choisissant "Return" pour le champ "Equiv." dans le menu déroulant <no key > ; le bouton par défaut pulse automatiquement et prend la couleur par défaut

key equivalent modifier

Accès : lecture / écriture
Classe : *number*

La touche de fonction du raccourci clavier ; vous pouvez régler cette propriété sur la touche Commande ou Option, ou sur les deux en même temps, dans la fenêtre Info d'Interface Builder ; toutefois, la valeur retournée par cette propriété est un nombre ; par défaut 0

roll over

Accès : lecture / écriture
Classe : *boolean*
Le bouton se comporte-t-il comme un roll over ?

sound

Accès : lecture / écriture
Classe : *sound* (page 70)
Le son produit lorsque le bouton est cliqué

state

Accès : lecture / écriture
Classe : *une des constantes* de cell state value (page 178)
L'état du bouton ; voir aussi la propriété *allows mixed state*

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre du bouton ; voir aussi la section "Discussion"

transparent

Accès : lecture / écriture
Classe : *boolean*
Le bouton est-il transparent ?

Éléments des objets de la classe Button

Un objet button peut contenir uniquement les éléments qu'il hérite de control (page 280).

Commandes supportées par les objets de la classe Button

Votre script peut envoyer la commande suivante à un objet button :

`highlight` (page 335)

Events supportés par les objets de la classe Button

Un objet button supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

AppleScript Studio permet d'accéder à plusieurs types d'objets button et les applications distribuées avec AppleScript Studio comportent de nombreux exemples de travail avec ces objets. L'instruction suivante montre une partie de la terminologie de base utilisable avec les boutons.

Dans la plupart des cas, vous utiliserez un bouton pour déclencher une action dans un gestionnaire clicked (page 348). Le gestionnaire clicked suivant, extrait de l'application "Currency Converter" distribuée avec AppleScript Studio, exécute juste la conversion monétaire, basée sur les valeurs saisies par l'utilisateur, et affiche le résultat. Le paramètre `theObject` du gestionnaire clicked se réfère au bouton. Dans ce cas, le gestionnaire utilise la fenêtre du bouton pour obtenir d'autres objets.

```
on clicked theObject
  tell window of theObject
    try
      set theRate to contents of text field "rate"
      set theAmount to contents of text field "amount" as number
      set contents of text field "total" to theRate * theAmount
    on error
      set contents of text field "total" to 0
    end try
  end tell
end clicked
```

L'instruction suivante désactive un bouton nommé "someButton". Notez que la propriété *enabled* n'est pas propre aux objets button, mais elle est héritée de la classe control (page 280).

```
set enabled of button someButton to "false"
```

Les instructions suivantes, extraites de l'application "Unit Converter" distribuée avec AppleScript Studio, montrent comment déterminer si un bouton particulier d'une view, comportant plusieurs boutons, fut la cible. Ici, le bouton en question (le bouton "Convert") fait partie d'un objet box (page 197).

```
on clicked theObject
  tell window "Main"
```

```
    if theObject is equal to button "Convert" of box 1 then
      my convert()
    else if ...
      ...
  end tell
end clicked
```

Discussion

Les boutons fournissent un mécanisme simple pour zapper leur titre ou icone lorsque l'utilisateur alterne leur état. Par exemple, pour créer dans Interface Builder un bouton qui alterne son texte entre "Start" et "Stop", vous suivrez ces étapes :

1. Glissez un objet button du panneau "Cocoa-Controls" sur la fenêtre visée. Vous pouvez utiliser n'importe quel modèle de bouton affichant NSButton lorsque vous laissez le curseur de la souris dessus.
2. Avec le bouton sélectionné dans la fenêtre cible, ouvrez la fenêtre Info en choisissant "Show Info" dans le menu "Tools" ou appuyez sur les touches Cmd + Maj. + I.
3. Si le menu déroulant "Behavior" du panneau Attributes de la fenêtre Info n'est pas actif, choisissez un type de bouton dans le menu déroulant "Type" qui provoque son activation, soit "Rounded Bevel Button", "Square Button" ou "Round Button".
4. Réglez le menu déroulant "Behavior" sur "Toggle".
5. Saisissez "Start" dans le champ "Title" et "Stop" dans le champ "Alt. Title".
6. Pour tester le bouton, choisissez "Test Interface" dans le menu "File" (ou appuyez sur Cmd + R). Vous pouvez à présent tester réellement votre bouton et zapper son titre entre "Start" et "Stop". Pour revenir en mode création, quittez l'application en choisissant "Quitter newApplication" du menu "Interface Builder" ou appuyez sur Cmd + Q.

Le panneau Attributes fournit d'autres réglages, y compris des champs réglant l'icone et l'icone alternée des boutons affichant une icone. Notez que vous pouvez afficher n'importe quelle image dans un bouton, pas uniquement un icone.

button cell

Pluriel : `button cells`
Hérite de : `cell` (page 264)
Classe Cocoa : NSButtonCell

Sous-classe de cell (page 264) qui implémente certains objets d'interface, comme les boutons poussoirs, les boutons case à cocher et les boutons radios. Vous n'aurez généralement pas besoin d'accéder aux propriétés d'un objet button cell, puisque vous pouvez accéder aux mêmes propriétés par l'intermédiaire de la classe button (page 254).

Dans Cocoa, un objet button cell peut être utilisé par n'importe quel zone d'une view qui est désignée pour envoyer un message à une cible lorsque cette zone est cliquée, bien qu'encore, ce ne soit pas une utilisation typique pour la plupart des applications AppleScript Studio. Pour des informations de même nature, voir les descriptions des classes action cell (page 254), cell (page 264) et button (page 254).

Vous pouvez créer et accéder dans Interface Builder à un objet button cell en suivant ces étapes :

1. Glissez un bouton du panneau "Cocoa-Controls" sur la fenêtre cible.
2. Sélectionnez le bouton.
3. Maintenez enfoncée la touche Option et avec le curseur de la souris, étirez une des poignées de redimensionnement. En même temps que vous faites glisser la souris, Interface Builder crée un objet matrix (page 289) contenant plusieurs objet button cell. Suivant si vous glissez à l'horizontal ou à la verticale, les objets button cell seront respectivement alignés à l'horizontal ou à la verticale.
4. Cliquer une seule fois sélectionnera l'objet matrix, double-cliquer sélectionnera un des objets button cell de l'objet matrix.

Propriétés des objets de la classe Button Cell

En plus des propriétés qu'il hérite de cell (page 264), un objet button cell possède ces propriétés :

alternate image

Accès : lecture / écriture
Classe : *image* (page 58)

L'image de la cellule lorsqu'elle est dans son état alterné ; voir la section "Discussion" de la classe button (page 254)

alternate title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de la cellule lorsqu'elle est dans son état alterné ; voir la section "Discussion" de la classe button (page 254)

bezel style

Accès : lecture / écriture
Classe : *une des constantes* de bezel style (page 175)
Le style de contour de la cellule

button type

Accès : lecture / écriture
Classe : *une des constantes* de button type (page 177)
Le type de bouton de la cellule

highlights by

Accès : lecture / écriture
Classe : *integer (énumération ; équivalent de "Behavior" pour un bouton dans Interface Builder)*
Le mécanisme par lequel le bouton est illuminé ; la plupart des applications n'auront pas besoin de travailler avec cette propriété dans leurs scripts et il n'y a pas (dans la version 1.3 d'AppleScript Studio) de constantes AppleScript Studio définies pour l'évaluer ; toutefois, vous pouvez lire la description de la classe NSCell dans la documentation Cocoa ; vous pouvez régler le comportement du bouton dans Interface Builder en utilisant les menus déroulants "Type" et "Behavior" du panneau Attributes de la fenêtre Info

image dims when disabled

Accès : lecture / écriture
Classe : *boolean*
L'image est-elle estompée lorsque l'objet button cell est indisponible ?

key equivalent modifier

Accès : lecture / écriture
Classe : *integer*
La touche de fonction du raccourci clavier ; voir la description de cette

propriété dans la classe button (page 254)

roll over

Accès : lecture / écriture
Classe : *boolean*
L'objet button cell se comporte-t-il comme un roll over ?

shows state by

Accès : lecture / écriture
Classe : *integer*
Non supportée dans la version 1.3 d'AppleScript Studio ; la façon dont l'objet button cell montre son état

sound

Accès : lecture / écriture
Classe : *sound* (page 70)
Le son joué par l'objet button cell lorsqu'il est cliqué

transparent

Accès : lecture / écriture
Classe : *boolean*
L'objet button cell est-il transparent ?

Events supportés par les objets de la classe Button Cell

Un objet button cell supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Nib

`awake from nib` (page 123)

Exemples

Les applications AppleScript Studio scriptent généralement les objets button, plutôt que directement les objets button cell, et la classe button (page 254) a certaines propriétés identiques à celles de la classe button cell. De plus, la classe button (page 254) hérite de la classe control (page 280), laquelle possède une propriété *current cell* par l'intermédiaire de laquelle

vous pouvez accéder aux propriétés des objets button cell d'un bouton, si nécessaire. Par exemple, vous pouvez utiliser le script suivant dans l'application Éditeur de Scripts pour accéder à l'objet button cell d'un bouton poussoir de la fenêtre principale d'une application AppleScript Studio. Des instructions similaires fonctionneront dans le script d'une application AppleScript Studio (bien que vous n'aurez pas besoin de l'instruction `tell application`).

```
tell application "testApplication"
  -- check the "image dims when disabled" property:
  image dims when disabled of (current cell of first button of window 1)
    -- result: 1
  -- check transparency:
  transparent of (current cell of first button of window 1)
    -- result: 0
end tell
```

Version

La propriété *shows state by* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

cell

Pluriel : `cells`
Hérite de : `personne`
Classe Cocoa : NSCell

Fournit un mécanisme permettant d'afficher du texte ou des images dans une view sans le chapeautage de la sous-classe NSView.

Énormément utilisée par la plupart des classes control (page 280) pour implémenter leur fonctionnement interne. Les sous-classes cell incluent action cell (page 254), button cell (page 261), image cell (page 284) et text field cell (page 330).

Pour plus d'informations, voir Controls and Cells dans la documentation Cocoa.

Propriétés des objets de la classe Cell

Un objet cell possède ces propriétés :

alignment

Accès : lecture / écriture
Classe : *une des constantes* de text alignment (page 189)
L'alignement du texte de la cellule

allows editing text attributes

Accès : lecture / écriture
Classe : *boolean*
Les attributs de texte peuvent-ils être édités ?

allows mixed state

Accès : lecture / écriture
Classe : *boolean*
La cellule autorise-t-elle un état mixte ? Voir la description de cette propriété dans la classe button (page 254)

associated object

Accès : lecture / écriture
Classe : *item* (page 60)
L'objet associé avec la cellule

bezeled

Accès : lecture / écriture
Classe : *boolean*
La cellule a-t-elle un contour apparent ?

bordered

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle délimitée ?

cell size

Accès : lecture uniquement
Classe : *point*
La taille de la cellule ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; par exemple, {75, 19} indiquerait une largeur de 75 et une hauteur de 19 ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

cell type

Accès : lecture / écriture
Classe : *une des constantes* de cell type (page 179)
Le type de cellule

content

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu de la cellule ; synonyme de *contents*

contents

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu de la cellule ; synonyme de *content*

continuous

Accès : lecture / écriture
Classe : *boolean*
La cellule génère-t-elle des actions lorsqu'elle est appuyée ?

control size

Accès : lecture / écriture
Classe : *une des constantes* de control size (page 180)
La taille du contrôle de la cellule

control tint

Accès : lecture / écriture
Classe : *une des constantes* de control tint (page 180)
La couleur du contrôle de la cellule

control view

Accès : lecture uniquement
Classe : *control* (page 280)
Le contrôle que possède la cellule

double value

Accès : lecture / écriture
Classe : *real*
La valeur du contenu au format double ; 0.0 si le contenu ne peut pas être interprété au format double

editable

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle éditable ?

enabled

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle activée ?

entry type

Accès : lecture / écriture
Classe : *integer*
Le type d'entrée ; le type d'entrée est apprécié dans Cocoa, aussi vous ne devrez pas l'utiliser dans vos scripts

float value

Accès : lecture / écriture
Classe : *real*
La valeur du contenu au format décimal ; 0.0 si le contenu ne peut pas être interprété au format décimal

font

Accès : lecture / écriture
Classe : *font* (page 54)
La police de la cellule

formatter

Accès : lecture / écriture
Classe : *formatter* (page 56)
Le "formatter" de la cellule ; cette propriété n'est pas supportée dans la version 1.3 d'AppleScript Studio ; toutefois, voir la section "Exemples" de la classe formatter (page 56) pour une description sur la manière d'utiliser la commande call method (page 94) pour obtenir le "formatter" et en tirer des informations

has valid object value

Accès : lecture uniquement
Classe : *boolean*
La cellule contient-elle une valeur valide ? Un objet valide est un objet que le "formatter" de la cellule (s'il est présent) peut comprendre

highlighted

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle illuminée ?

image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image de la cellule

image position

Accès : lecture / écriture
Classe : *une des constantes* de cell image position (page 177)
La position de l'image dans la cellule

imports graphics

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les graphiques soient importés ?

integer value

Accès : lecture / écriture
Classe : *integer*
La valeur du contenu au format entier ; 0 si le contenu ne peut pas être interprété au format entier

key equivalent

Accès : lecture uniquement
Classe : *Unicode text*
Le raccourci clavier de la cellule ; voir la description de cette propriété dans la classe button (page 254)

menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le contexte du menu de la cellule, s'il y a

mouse down state

Accès : lecture uniquement
Classe : *integer*
L'état de la souris lorsqu'elle a été cliqué dans la cellule

next state

Accès : lecture / écriture
Classe : *integer*
L'état suivant de la cellule

opaque

Accès : lecture uniquement
Classe : *boolean*
La cellule est-elle opaque ?

scrollable

Accès : lecture / écriture
Classe : *boolean*
La cellule peut-elle être scrollée ?

selectable

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle sélectionnable ?

sends action when done editing

Accès : lecture / écriture
Classe : *boolean*
La cellule doit-elle envoyer ses actions lorsqu'elle a fini ? Les applications Cocoa connectent généralement les objets d'interface aux méthodes de l'objet cible, mais les applications AppleScript Studio les connectent aux gestionnaires d'Events du script ; toutefois, vous ne pouvez pas connecter de gestionnaires d'Events à un objet cell

state

Accès : lecture / écriture
Classe : *une des constantes* de cell state value (page 178)
L'état de la cellule

string value

Accès : lecture / écriture
Classe : *Unicode text*
La valeur du contenu de la cellule au format texte

tag

Accès : lecture / écriture
Classe : *integer*

L'étiquette de la cellule

target

Accès : lecture / écriture
Classe : *item* (page 60)
La cible des actions de la cellule ; les applications Cocoa connectent généralement les objets d'interface aux méthodes de l'objet cible, mais les applications AppleScript Studio les connectent aux gestionnaires d'Events du script ; toutefois, vous ne pouvez pas connecter de gestionnaires d'Events à un objet cell

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de la cellule

wraps

Accès : lecture / écriture
Classe : *boolean*
La cellule insère-t-elle des retours automatiques en fin de ligne ?

Commandes supportées par les objets de la classe Cell

Votre script peut envoyer la commande suivante à un objet cell :

`perform action` (page 337)

Events supportés par les objets de la classe Cell

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Vous scripterez généralement une sous-classe de control (page 280) ou view (page 229) contenant un objet cell (ou une sous-classe de cell), pas l'objet cell lui-même. L'unique situation où vous devrez accéder à l'objet cell lui-même est lorsque vous travaillerez avec un objet matrix (page 289). Un objet matrix est utilisé pour créer un groupe d'objets cell, comme des boutons radio.

L'application "Assistant", distribuée avec AppleScript Studio depuis la version 1.2, utilise un objet matrix avec des boutons radio pour spécifier la

gravité d'un problème. Elle utilise l'instruction suivante dans son gestionnaire `updateValue` pour obtenir la propriété *title* du bouton radio actuellement sélectionné. Cette instruction règle la propriété *severity* (créée pour l'occasion) sur le titre de la cellule en cours de l'objet matrix (l'objet matrix est aussi nommé "severity") :

```
set my severity to title of current cell of matrix "severity"
```

Version

La propriété *content* est apparue avec la version 1.2 d'AppleScript Studio. Vous pouvez utiliser au choix *content* et *contents*, sauf à l'intérieur d'un gestionnaire d'Events, `contents of theObject` retournant une référence à l'objet plutôt que son contenu courant. Pour obtenir dans un gestionnaire d'Events le contenu d'un objet (comme le texte contenu dans un text field (page 325)), vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

La propriété *formatter* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Pour un exemple de script montrant la différence entre `content` et `contents`, voir la section "Version" de la classe control (page 280).

color well

Pluriel : `color wells`
Hérite de : `control` (page 280)
Classe Cocoa : NSColorWell

Supporte la sélection et l'affichage d'une couleur. Un objet color-panel (page 514) utilise un objet color well pour afficher la couleur courante sélectionnée. Vous travaillerez généralement avec les couleurs par l'intermédiaire de la propriété *color panel* de la classe application (page 29), pas directement avec un objet color well.

L'illustration 4.2 montre un objet color well dans une fenêtre. Cliquez sur cet objet ouvre le panel couleurs ; choisir une couleur dans le panel couleurs règle la couleur de l'objet color well.

Vous trouverez l'objet color well dans le panneau "Cocoa-Controls" d'Interface Builder. Vous pouvez régler les attributs d'un objet color well dans la fenêtre Info d'Interface Builder.

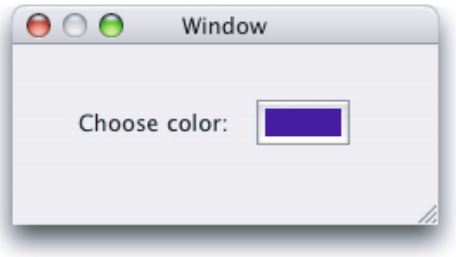

FIG. 4.2 - *Un objet color well dans une fenêtre*

Pour des informations de même nature, voir Using Color dans la documentation Cocoa.

Propriétés des objets de la classe Color Well

En plus des propriétés qu'il hérite de control (page 280), un objet color well possède ces propriétés :

active

Accès : lecture / écriture
Classe : *boolean*
L'objet color well est-il actif ?

bordered

Accès : lecture / écriture
Classe : *boolean*
L'objet color well a-t-il une bordure ?

color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de l'échantillon ; une liste de trois nombres entiers représentant chacun un composant de la couleur ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0}

Events supportés par les objets de la classe Color Well

Un objet color well supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)
Cliquer sur un objet color well ouvre automatiquement un objet color-panel (page 514), et votre gestionnaire `clicked` sera appelé uniquement la première fois que l'objet color well sera cliqué.

Glisser-Déposer

`conclude drop` (page 481)

`drag` (page 483)

`drag entered` (page 483)

`drag exited` (page 484)

`drag updated` (page 485)

`drop` (page 486)

`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)

`keyboard up` (page 134)

Souris

`mouse down` (page 137)

`mouse dragged` (page 138)

`mouse entered` (page 139)

`mouse exited` (page 139)

`mouse up` (page 141)

`right mouse down` (page 147)

`right mouse dragged` (page 149)

`right mouse up` (page 150)

`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Vous travaillerez en général avec les couleurs par l'intermédiaire de la propriété *color panel* de la classe application (page 29), pas directement avec un objet color well. Pour des exemples, voir color-panel (page 514).

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

combo box

Pluriel : `combo boxes`
Hérite de : `text field` (page 325)
Classe Cocoa : NSComboBox

Un contrôle qui fournit deux façons de saisir une valeur : soit directement par l'intermédiaire d'un champ de saisie (comme un champ texte), soit en choisissant une valeur dans une liste de valeurs pré-sélectionnées d'un menu déroulant.

L'illustration 4.3 montre un objet combo box avec sa liste cachée.

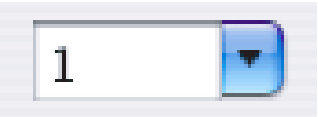

FIG. 4.3 - *Un objet combo box sans sa liste visible*

L'illustration 4.4 montre le même combo box mais avec la liste déroulée.

Vous trouverez l'objet combo box dans le panneau "Cocoa-Text" d'Interface Builder. Vous pouvez régler beaucoup d'attributs des objets combo box dans la fenêtre Info d'Interface Builder. Notez que dans la version 1.3 d'AppleScript Studio, vous ne pouvez pas utiliser un objet data source (page 386) avec un objet combo box.

Pour plus d'informations, voir Combo Boxes dans la documentation Cocoa.

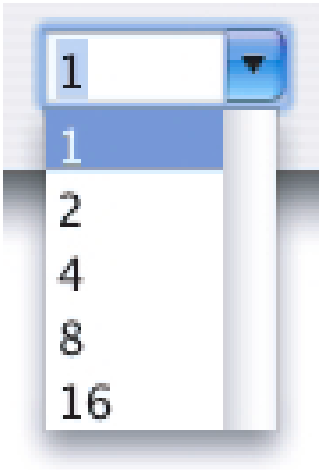

FIG. 4.4 - *Un objet combo box avec sa liste déroulée*

Propriétés des objets de la classe Combo Box

En plus des propriétés qu'il hérite de text field (page 325), un objet combo box possède ces propriétés :

auto completes

Accès : lecture / écriture
Classe : *boolean*
Le combo box utilise-t-il le remplissage automatique lors de la saisie ? Par défaut, cette propriété vaut `false` ; elle peut être réglée dans la fenêtre Info d'Interface Builder

current item

Accès : lecture / écriture
Classe : *integer*
L'index de l'élément courant, base 0

data source

Accès : lecture / écriture
Classe : *data source* (page 386)
Non supportée dans la version 1.3 d'AppleScript Studio ; la data source de l'objet combo box ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder, bien que la meilleure façon soit de la régler dans un script, comme le montrent les sections "Exemples" de la commande append (page 417) et de la classe data item (page 379).

has vertical scroller

Accès : lecture / écriture
Classe : *boolean*
Le combo box a-t-il un ascenseur vertical ? Par défaut, cette propriété vaut `true` ; se règle avec “Scrollable” dans la fenêtre Info d’Interface Builder

intercell spacing

Accès : lecture / écriture
Classe : *list*
L’espace horizontal et vertical entre les cellules de la liste du combo box ; exprimée sous forme d’une liste de deux nombres ; par défaut, cette propriété vaut {3, 2}

item height

Accès : lecture / écriture
Classe : *real*
La hauteur d’un élément

uses data source

Accès : lecture / écriture
Classe : *boolean*
Non supportée dans la version 1.3 d’AppleScript Studio ; le combo box utilise-t-il une data source pour ces éléments ? Vous pouvez régler cette propriété dans la fenêtre Info d’Interface Builder

Éléments des objets de la classe Combo Box

En plus des éléments qu’il hérite de la classe text field (page 325), un objet combo box peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans “Les formes-clés standards” (page 15).

`combo box item` (page 279)
spécifier par : “Les formes-clés standards” (page 15)
représente les éléments de la liste du menu déroulant de l’objet combo box ; stockés sous forme d’une liste d’éléments texte

Commandes supportées par les objets de la classe Combo Box

Votre script peut envoyer la commande suivante à un objet combo box :

`scroll` (page 339)

Events supportés par les objets de la classe Combo Box

Un objet combo box supporte les gestionnaires répondant aux Events suivants :

Action

`action` (page 345)

Combo Box

`selection changed` (page 351)
`selection changing` (page 352)
`will dismiss` (page 355)
`will pop up` (page 356)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Edition

`begin editing` (page 346)
`changed` (page 348)
`end editing` (page 350)
`should begin editing` (page 353)
`should end editing` (page 354)

Clavier

`keyboard up` (page 134)

Souris

`mouse entered` (page 139)
`mouse exited` (page 139)

scroll wheel (page 151)

Nib

awake from nib (page 123)

View

bounds changed (page 243)

Exemples

Vous pouvez accéder aux éléments de la liste du menu déroulant d'un objet combo box par l'intermédiaire de sa propriété *combo box item*. Étant donné un objet combo box nommé "combo" dans la fenêtre en avant-plan, l'instruction suivante retournera la liste des éléments texte, chaque élément représentant un élément de la liste du menu déroulant de cet objet combo box :

```
every combo box item of combo box "combo" of window 1
```

Les lignes suivantes supprimeront tous les éléments du combo box et ajouteront un nouvel élément :

```
delete every combo box item of combo box "combo" of window 1
make new combo box item at end of combo box items of combo box "combo" of
  window 1 with data "Test Item"
```

Vous pouvez supprimer un objet combo box item avec son index, comme avec par exemple l'instruction suivante :

```
delete combo box item 2 of combo box 1 of window 1
```

Pour obtenir le contenu d'un objet combo box (la valeur de la partie texte du combo box), vous pouvez utiliser cette instruction :

```
set comboContents to contents of combo box "combo" of window "main"
```

Voir aussi la section "Exemples" de la classe combo box item (page 279).

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

Les propriétés *data source* et *uses data source* de cette classe ne sont pas supportées dans la version 1.3 d'AppleScript Studio.

combo box item

Pluriel : `combo boxes items`
Hérite de : `personne`
Classe Cocoa : ASKComboBoxItem

Représente un élément de la liste du menu déroulant d'un objet combo box.

Pour plus d'informations ,voir combo box (page 274).

Events supportés par les objets de la classe Combo Box Item

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires d'Events.

Exemples

Étant donné un objet combo box nommé "combo" dans la fenêtre en avant-plan, l'instruction suivante retournera la liste des éléments texte, chaque élément représentant un élément de la liste du menu déroulant de cet objet combo box :

```
every combo box item of combo box "combo" of window 1
```

Voir la section "Exemples" de la classe combo box (page 274) pour plus d'informations sur la création et la suppression des objets combo box item.

Version

La classe combo box item est apparue avec la version 1.1 d'AppleScript Studio.

control

Pluriel : controls
Hérite de : view (page 229)
Classe Cocoa : NSControl

Une super-classe-résumé fournissant trois caractéristiques fondamentales pour l'implémentation des dispositifs d'interface utilisateur (comme les boutons, les champs textes, les ascenseurs, etc) : dessinement des dispositifs d'affichage, répondre aux Events de l'utilisateur et envoyer les messages actions.

Fonctionne de près avec les objets cell (page 264). La plupart des applications n'auront souvent pas besoin de scripter directement un objet control, mais plutôt de scripter des sous-classes, comme button (page 254) et text field (page 325).

Pour plus d'informations, voir Controls and Cells dans la documentation Cocoa.

Propriétés des objets de la classe Control

En plus des propriétés qu'il hérite de view (page 229), un objet control possède ces propriétés :

alignment

Accès : lecture / écriture
Classe : *une des constantes* de text alignment (page 189)
Le type d'alignement du texte de l'objet control

cell

Accès : lecture / écriture
Classe : *cell* (page 264)
La cellule de l'objet control

content

Accès : lecture / écriture
Classe : *item* (page 60)
La valeur de l'objet control ; synonyme de *contents*

contents

Accès : lecture / écriture

Classe : *item* (page 60)
La valeur de l'objet control ; synonyme de *content*

continuous

Accès : lecture / écriture
Classe : *boolean*
L'objet control génère-t-il continuellement des actions ?

current cell

Accès : lecture uniquement
Classe : *cell* (page 264)
La cellule en cours de l'objet control

current editor

Accès : lecture / écriture
Classe : *text* (page 569) ou *text view* (page 569)
L'éditeur courant si l'objet est sur le point d'être édité, si ce n'est pas le cas, retourne rien ; généralement retourne une valeur comme `current field editor of window id 1 of application "StudioTest"` ; voir la propriété *field editor* de la classe text (page 569) pour une description d'un éditeur

double value

Accès : lecture / écriture
Classe : *real*
La valeur de l'objet control au format double ; 0.0 si le contenu ne peut pas être interprété au format double

enabled

Accès : lecture / écriture
Classe : *boolean*
L'objet control est-il activé ?

float value

Accès : lecture / écriture
Classe : *real*
La valeur de l'objet control au format décimal ; 0.0 si le contenu ne peut pas être interprété au format décimal

font

Accès : lecture / écriture

Classe : *font* (page 54)
La police de l'objet control

formatter

Accès : lecture / écriture
Classe : *formatter* (page 56)
Le "formatter" de l'objet control ; cette propriété n'est pas supportée dans la version 1.3 d'AppleScript Studio ; toutefois, voir la section "Exemples" de la classe formatter (page 56) pour une description sur la manière d'utiliser la commande call method (page 94) pour obtenir le "formatter" et en tirer des informations

ignores multiple clicks

Accès : lecture / écriture
Classe : *boolean*
L'objet control ignore-t-il les multiples clics ?

integer value

Accès : lecture / écriture
Classe : *integer*
La valeur de l'objet control au format entier ; 0 si le contenu ne peut pas être interprété au format entier

string value

Accès : lecture / écriture
Classe : *Unicode text*
La valeur de l'objet control au format texte

target

Accès : lecture / écriture
Classe : *item* (page 60)
La cible des actions du control

Éléments supportés par les objets de la classe Control

Un objet control peut uniquement contenir les éléments qu'il hérite de la classe view (page 229).

Commandes supportées par les objets de la classe Control

Votre script peut envoyer la commande suivante à un objet control :

`perform action` (page 337)

Events supportés par les objets de la classe Control

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

La classe control est une classe-résumé que vous ne viserez généralement pas dans vos scripts, mais certainement des sous-classes, comme button (page 254) ou slider (page 315).

Version

La propriété *content* est apparue avec la version 1.2 d'AppleScript Studio. Vous pouvez utiliser au choix *content* et *contents*, sauf à l'intérieur d'un gestionnaire d'Events, `contents of theObject` retournant une référence à l'objet plutôt que son contenu courant. Pour obtenir dans un gestionnaire d'Events le contenu d'un objet (comme le texte contenu dans un text field (page 325)), vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

La propriété *formatter* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Les instructions suivantes d'un gestionnaire `action` utilise la commande log pour montrer le résultat de l'utilisation de *content* et *contents* sur un champ texte contenant le texte "Some text".

```
on action theObject
  log theObject
    -- result: text field 1 of window 1
  log contents of theObject
    -- result: text field 1 of window 1
  log contents of contents of theObject as string
    -- result: "Some text"
  log content of theObject as string
    -- result: "Some text"
end action
```

image cell

Pluriel : `image cells`
Hérite de : `cell` (page 264)
Classe Cocoa : NSImageCell

Affiche une image dans un cadre.

Fournit des propriétés pour spécifier le type de cadre et, l'alignement et l'échelle de l'image. Un objet image cell est généralement associé avec un type d'objet de la classe control (page 280), comme un objet image view (page 285), matrix (page 289) ou table view (page 403).

Vous pouvez créer et accéder dans Interface Builder à un objet image cell en suivant ces étapes :

1. Glissez un image view du panneau "Cocoa-Controls" sur la fenêtre visée.
2. Sélectionnez l'image view.
3. Maintenez enfoncée la touche Option et avec le curseur de la souris, étirez une des poignées de redimensionnement. En même temps que vous faites glisser la souris, Interface Builder crée un objet matrix (page 289) contenant plusieurs objet image cell. Suivant si vous glissez à l'horizontal ou à la verticale, les objets image cell seront respectivement alignés à l'horizontal ou à la verticale.
4. Cliquer une fois sélectionnera l'objet matrix ; double-cliquer sélectionnera un des objets image cell de l'objet matrix.

Pour plus d'informations, voir image (page 58), ainsi que Image Views, Matrices et Table Views dans la documentation Cocoa.

Propriétés des objets de la classe Image Cell

En plus des propriétés qu'il hérite de cell (page 264), un objet image cell possède ces propriétés :

image alignment

Accès : lecture / écriture
Classe : *une des constantes* de image alignment (page 183)
L'alignement de l'image de la cellule

image frame style

Accès : lecture / écriture
Classe : *une des constantes* de image frame style (page 184)
Le type de cadre de l'image

image scaling

Accès : lecture / écriture
Classe : *une des constantes* de image scaling (page 184)
L'échelle de l'image

Events supportés par les objets de la classe Image Cell

Un objet image cell supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Nib

`awake from nib` (page 123)

Exemples

Vous ne scripterez généralement pas un objet image cell. À la place, vous pouvez scripter les mêmes propriétés d'un objet image view (page 285) (*image frame style*, *image alignment* et *image scaling*).

`image view`

Pluriel : `image views`
Hérite de : `control` (page 280)
Classe Cocoa : NSImageView

Affiche une image dans un cadre, et peut occasionnellement autoriser un utilisateur à glisser une image dessus.

L'illustration 4.5 montre une image affichée dans un objet image view par l'application "Image" distribuée avec AppleScript Studio.

Vous pouvez stocker une image dans votre projet AppleScript Studio en glissant un fichier image depuis le Finder dans un des groupes de la liste

"Files" du panneau "Groups & Files" de Xcode, ou en utilisant le menu "Add Files..." du menu "Project". Vous pouvez aussi glisser les images sur le panneau "Images" de la fenêtre Nib dans Interface Builder. Vous pouvez afficher une image dans un objet image view en la chargeant avec la commande load image (page 100). Pour des informations de même nature, voir la classe image (page 58), ainsi que Images Views dans la documentation Cocoa.

Si vous chargez continuellement des images et que vous ne les libérez pas, la mémoire utilisée par votre application va augmenter. Pour des informations sur la manière de libérer des objets image (page 58), movie (page 63) ou sound (page 70), voir la section "Discussion" de la commande load image (page 100).

FIG. 4.5 - *Une fenêtre affichant une image dans un objet image view (extrait de l'application "Image")*

Propriétés des objets de la classe Image View

En plus des propriéts qu'il hérite de control (page 280), un objet image view possède ces propriétés :

editable

Accès : lecture / écriture
Classe : *boolean*

L'image view est-elle éditable ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image de la view ; vous pouvez régler cette propriété dans Interface Builder en glissant-déposant une image sur l'objet image view

image alignment

Accès : lecture / écriture
Classe : *une des constantes* de image alignment (page 183)
L'alignement de l'image ; par défaut, cette propriété vaut `center alignment` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

image frame style

Accès : lecture / écriture
Classe : *une des constantes* de image frame style (page 184)
Le type de cadre de l'image ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

image scaling

Accès : lecture / écriture
Classe : *une des constantes* de image scaling (page 184)
L'échelle de l'image ; par défaut, cette propriété vaut `scale proportionally` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

Éléments des objets de la classe Image View

Un objet image view peut uniquement contenir les éléments qu'il hérite de la classe control (page 280).

Events supportés par les objets de la classe Image View

Un objet image view supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Le gestionnaire awake from nib (page 123) suivant est extrait de l'application "Image" distribuée avec AppleScript Studio. Le gestionnaire utilise simplement la commande load image (page 100) pour charger une image du projet (l'image est stockée dans le groupe "Resources" du projet et elle est nommée "AboutBox.tiff").

```
on awake from nib theObject
  set image of image view "image" of window "main" to load image "AboutBox"
end awake from nib
```

Si vous chargez continuellement des images et que vous ne les libérez pas, la mémoire utilisée par votre application va augmenter. Pour des informations sur la manière de libérer des objets image (page 58), movie (page 63) ou sound (page 70), voir la section “Discussion” de la commande load image (page 100).

matrix

Pluriel : matrices
Hérite de : control (page 280)
Classe Cocoa : NSMatrix

Utilisée pour créer des groupes d’objets cell (page 264), comme des boutons radios, fonctionnant ensemble de différentes manières.

L’illustration 4.6 montre un objet matrix contenant trois boutons radios.

FIG. 4.6 - *Un objet matrix avec trois boutons radios*

Vous trouverez l’objet matrix (contenant des boutons radios) dans le panneau “Cocoa-Controls” d’Interface Builder. Vous pouvez régler beaucoup d’attributs des objets matrix dans la fenêtre Info d’Interface Builder. Pour des informations sur la création d’objets matrix contenant d’autres sortes d’objets, voir la classe scroll view (page 213), ainsi que les classes button cell

(page 261), image cell (page 284), secure text field cell (page 314) et text field cell (page 330).

Pour plus d'informations, voir aussi Matrices dans la documentation Cocoa.

Propriétés des objets de la classe Matrix

En plus des propriétés qu'il hérite de control (page 280), un objet matrix possède ces propriétés :

allows empty selection

Accès : lecture / écriture
Classe : *boolean*
L'objet matrix autorise-t-il une sélection vide ? Par défaut, cette propriété vaut `false`, par exemple, pour une matrice de boutons radios ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

auto scroll

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet matrix défile automatiquement ?

auto sizes cells

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet matrix dimensionne automatiquement ses cellules ? Par défaut, cette propriété vaut `false` pour une matrice de boutons radios ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond de l'objet matrix ; exprimée sous forme d'une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur rouge peut être représentée par {65535, 0, 0} ; par défaut, cette propriété vaut {65535, 65535, 65535}, ou la couleur blanche ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

cell background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond des cellules de l'objet matrix ; exprimée sous forme d'une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par défaut, cette propriété vaut {65535, 65535, 65535}, ou la couleur blanche

cell size

Accès : lecture / écriture
Classe : *point*
La taille de chaque cellule de l'objet matrix ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées ; non supportée dans la version 1.3 d'AppleScript Studio ; toutefois, vous pouvez utiliser la commande call method (page 94) pour obtenir ou régler cette propriété ; la première instruction ci-dessous obtient la taille — la seconde la règle :

```
set cellSize to call method "cellSize" of matrix 1 of window 1
call method "setCellSize:" of matrix 1 of window 1
  with parameter {200, 20}
```

current cell

Accès : lecture / écriture
Classe : *cell* (page 264)
La cellule courante

current column

Accès : lecture / écriture
Classe : *integer*
La colonne courante de l'objet matrix

current row

Accès : lecture / écriture
Classe : *integer*
La rangée courante de l'objet matrix

draws background

Accès : lecture / écriture
Classe : *boolean*

Faut-il que l'objet matrix dessine son fond ? Par défaut, cette propriété vaut `false` pour une matrice de boutons radios ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

draws cell background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les cellules de l'objet matrix dessinent leur fond ?

intercell spacing

Accès : lecture / écriture
Classe : *list*
L'espace vertical et horizontal entre les cellules de l'objet matrix ; par défaut, les deux valeurs sont égales à 1.0 dans le système de coordonnées de l'objet matrix (voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées) ; non supportée dans la version 1.3 d'AppleScript Studio ; toutefois, vous pouvez utiliser la commande call method (page 94) pour obtenir ou régler cette propriété ; la première instruction ci-dessous obtient l'espace — la seconde le règle :

```
set spacing to call method "intercellSpacing" of matrix 1 of window 1
call method "setIntercellSpacing:" of matrix 1 of window 1
  with parameter {2.0, 2.0}
```

key cell

Accès : lecture / écriture
Classe : *cell* (page 264)
La cellule-clé de l'objet matrix

matrix mode

Accès : lecture / écriture
Classe : *une des constantes* de matrix mode (page 185)
Le mode de l'objet matrix (par exemple, `radio mode`)

next text

Accès : lecture / écriture
Classe : *n'importe*
Non supportée dans la version 1.3 d'AppleScript Studio ; se servir de la méthode de la classe NSMatrix sur laquelle cette propriété est basée n'est pas encouragée, aussi cette propriété risque de ne jamais être

supportée ; le prochain éditeur de l'objet matrix ; voir la propriété *field editor* de la classe text (page 569) pour une description d'un éditeur

previous text

Accès : lecture / écriture
Classe : *n'importe*
Non supportée dans la version 1.3 d'AppleScript Studio ; se servir de la méthode de la classe NSMatrix sur laquelle cette propriété est basée n'est pas encouragée, aussi cette propriété risque de ne jamais être supportée ; le précédent éditeur de l'objet matrix ; voir la propriété *field editor* de la classe text (page 569) pour une description d'un éditeur

prototype cell

Accès : lecture / écriture
Classe : *cell* (page 264)
Le prototype de cellule de l'objet matrix

scrollable

Accès : lecture / écriture
Classe : *boolean*
L'objet matrix est-il défilable ? Non supportée dans la version 1.3 d'AppleScript Studio ; toutefois, vous pouvez utiliser la commande call method (page 94) pour régler cette propriété (mais pas pour l'obtenir) ; notez que vous devrez transmettre une valeur booléenne à la commande call method sous forme d'une liste à un seul élément :

```
call method "setScrollable:" of matrix 1 of window 1
  with parameters {true}
```

selection by rect

Accès : lecture / écriture
Classe : *boolean*
Les cellules peuvent-elles être sélectionnées par le rectangle ? Vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

tab key traverses cells

Accès : lecture / écriture
Classe : *boolean*
La touche tabulation peut-elle servir à passer d'une cellule à l'autre ?

Éléments des objets de la classe Matrix

En plus des éléments qu'il hérite de la classe control (page 280), un objet matrix peut contenir les éléments listés ci-dessous. Votre script peut spécifier la plupart des éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`cell` (page 264)
spécifier par : "Les formes-clés standards" (page 15)
les cellules de l'objet matrix

Events supportés par les objets de la classe Matrix

Un objet matrix supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)

`right mouse dragged` (page 149)

`right mouse up` (page 150)

`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

L'application "Drawer" distribuée avec AppleScript Studio utilise un objet matrix, contenant quatre boutons radios, pour spécifier le côté sur lequel son tiroir devra s'ouvrir (gauche, haut, droite ou bas). Les instructions suivantes, extraites du gestionnaire awake from nib (page 123) de cette application, règle la propriété *current row* de l'objet matrix pour indiquer le bouton radio sélectionné, se basant sur les informations de la propriété *edge* de l'objet drawer.

```
on awake from nib theObject
  tell theObject
    set openOnEdge to edge of drawer "drawer"
    ...
    if openOnEdge is left edge then
      set current row of matrix "open on" to 1
    else if openOnEdge is top edge then
      set current row of matrix "open on" to 2
    else if openOnEdge is right edge then
      set current row of matrix "open on" to 3
    else if openOnEdge is bottom edge then
      set current row of matrix "open on" to 4
    end if
    ...
  end tell
end awake from nib
```

Les instructions suivantes, extraites du gestionnaire clicked (page 348) de l'application "Drawer", montre comment extraire des informations sur

les rangées de l'objet matrix, de façon à ce que lorsque le bouton "drawer" est cliqué, l'application sache sur quel côté ouvrir le tiroir.

```
on clicked theObject
  tell window "main"
    if theObject is equal to button "drawer" then
      ...
      set openOnSide to current row of matrix "open on"
  ...
```

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

Les propriétés *cell size*, *intercell spacing*, *next text*, *previous text* et *scrollable* de cette classe ne sont pas supportées dans la version 1.3 d'AppleScript Studio.

movie view

Pluriel : `movie views`
Hérite de : `view` (page 229)
Classe Cocoa : NSMovieView

Affiche un film dans un cadre et fournit des propriétés associées à la lecture et à l'édition du film.

Vous utiliserez la commande load movie (page 104) pour charger un objet movie (page 63) pour le movie view. Pour lire, avancer pas à pas ou se rendre à un endroit donné dans le film, vous utiliserez des commandes comme start (page 340), stop (page 343), play (page 338), step forward (page 342), step back (page 341) ou go (page 334) (pour se rendre à un emplacement donné).

L'illustration 4.7 montre un film lu dans un movie view dans l'application "Talking Head" distribuée avec AppleScript Studio.

Vous pouvez stocker un film dans votre projet AppleScript Studio en glissant un fichier film depuis le Finder sur un des groupes de la liste "Files" dans le panneau "Groups & Files" de Xcode, ou en utilisant le menu "Add Files..." du menu "Project". Vous pouvez afficher un film dans un movie view en le chargeant avec la commande load movie (page 104).

Vous pouvez insérer un objet movie view dans un fichier .nib dans Interface Builder en glissant l'objet représenté par le symbole QuickTime du panneau "Cocoa-GraphicsViews" sur la fenêtre visée.

FIG. 4.7 - *Un objet movie view (extrait de l'application "Talking Head")*

Propriétés des objets de la classe Movie View

En plus des propriétés qu'il hérite de view (page 229), un objet movie view possède ces propriétés :

controller visible

Accès : lecture / écriture
Classe : *boolean*
Le contrôleur est-il visible ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

editable

Accès : lecture / écriture
Classe : *boolean*
Le film est-il éditable ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

loop mode

Accès : lecture / écriture

Classe : *une des constantes* de quicktime movie loop mode (page 185)
Le type de boucle du lecteur ; par défaut, cette propriété vaut `normal` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

movie

Accès : lecture / écriture
Classe : *movie* (page 63)
Le film de la view à jouer ; vous chargerez le film dans vos scripts à l'aide de la commande load movie (page 104)

movie controller

Accès : lecture uniquement
Classe : *item* (page 60)
Le contrôleur de film QuickTime

movie file

Accès : lecture / écriture
Classe : *Unicode text*
Le chemin POSIX (délimité par des slashs) du fichier film du movie view ; voir la description de la classe ci-dessus pour des informations sur l'ajout de films à votre projet AppleScript Studio

movie rect

Accès : lecture / écriture
Classe : *bounding rectangle*
Les frontières du film dans la view ; une liste de quatre nombres {gauche, bas, droite, haut} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

muted

Accès : lecture / écriture
Classe : *boolean*
Le film est-il muet ? Par défaut, cette propriété vaut `false`

playing

Accès : lecture / écriture
Classe : *boolean*
Le film est-il lu ?

plays every frame

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le film lise chaque frame ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

plays selection only

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le lecteur lise uniquement la portion sélectionnée du film ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

rate

Accès : lecture / écriture
Classe : *real*
La vitesse à laquelle le film est lu

volume

Accès : lecture / écriture
Classe : *real*
Le volume sonore du film ; pour plus d'informations sur le volume sonore, voir la section "Exemples" de la classe slider (page 315)

Commandes supportées par les objets de la classe Movie View

Votre script peut envoyer les commandes suivantes à un objet movie view :

`copy` (de la Core suite Cocoa, décrite dans Core Suites de Scriptable Applications dans la documentation Cocoa)

`go` (page 334)

`play` (page 338)

`pause` (page 337)

`start` (page 340)

`step back` (page 341)

`step forward` (page 342)

`stop` (page 343)

Events supportés par les objets de la classe Movie View

Un objet movie view supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Le gestionnaire will open (page 165) suivant est extrait de l'application "Talking Head" distribuée avec AppleScript Studio. Le gestionnaire utilise

simplement la commande load movie (page 104) pour charger un film du projet lorsque la fenêtre contenant l'objet movie view est ouverte. L'image est stockée dans le groupe Resources du projet et elle est nommée "jumps.mov".

```
on will open theObject
  set movie of movie view "movie" of window "main" to load movie "jumps"
end will open
```

Si vous chargez continuellement des films et que vous ne les libérez pas, la mémoire utilisée par votre application va augmenter. Pour des informations sur la manière de libérer des objets image (page 58), movie (page 63) ou sound (page 70), voir la section "Discussion" de la commande load image (page 100).

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

popup button

Pluriel : `popup buttons`
Hérite de : `button` (page 254)
Classe Cocoa : NSPopUpButton

Fournit l'accès à un menu déroulant dans lequel un utilisateur peut choisir un élément.

L'illustration 4.8 montre un bouton popup affichant le premier élément de ce menu.

Vous trouverez l'objet popup button dans le panneau "Cocoa-Controls" d'Interface Builder. Vous pouvez régler les attributs des boutons popups dans la fenêtre Info d'Interface Builder.

Pour des informations de même nature, voir Application Menus and Pop-up Lists dans la documentation Cocoa.

Propriétés des objets de la classe Popup Button

En plus des propriétés qu'il hérite de la classe button (page 254), un objet popup button possède ces propriétés :

FIG. 4.8 - *Un bouton popup*

auto enables items

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les éléments du menu soit automatiquement activé ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

current menu item

Accès : lecture / écriture
Classe : *menu item* (page 499)
L'élément de menu actuellement choisi

preferred edge

Accès : lecture / écriture
Classe : *une des constantes* de rectangle edge (page 186)
Le côté favori pour présenter le menu sous réserve de sa position dans l'écran ; notez que le côté préféré sera ignoré s'il n'y a pas assez de place pour dérouler le menu dans cette direction

pulls down

Accès : lecture / écriture
Classe : *boolean*
La liste se déroule-t-elle sous le menu ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

Éléments des objets de la classe Popup Button

En plus des éléments qu'il hérite de la classe button (page 254), un objet de la classe popup button peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`menu` (page 495)
spécifier par : "Les formes-clés standards" (page 15)
les sous-menus de l'objet popup button

`menu item` (page 499)
spécifier par : "Les formes-clés standards" (page 15)
les éléments de menu de l'objet popup button

Commandes supportées par les objets de la classe Popup Button

Votre script peut envoyer la commande suivante à un objet popup button :

`synchronize` (page 343)

Events supportés par les objets de la classe Popup Button

Un objet popup button supporte les gestionnaires répondant aux Events suivants :

Action

`action` (page 345)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)

`keyboard up` (page 134)

Menu

`choose menu item` (page 505)
`will pop up` (page 356)

Souris

`mouse entered` (page 139)
`mouse exited` (page 139)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

L'instruction suivante, extraite du gestionnaire awake from nib (page 123) de l'application "Task List" (disponible depuis la version 1.2 d'AppleScript Studio), règle le titre d'un objet popup button nommé "priority" sur `"3"`.

```
set title of popup button "priority" to "3"
```

Vous supprimerez des éléments de menu dans un popup button avec la commande `delete` et vous en ajouterez avec la commande `make`. Par exemple, dans le fichier `Converter.applescript` de l'application "Unit Converter" distribuée avec AppleScript Studio, vous trouverez le gestionnaire `updateUnitTypes`. Ce gestionnaire montre comment supprimer tous les éléments d'un menu popup et les remplacer par de nouveaux lorsque vous avez besoin de mettre à jour ses éléments.

```
on updateUnitTypes()
  tell box 1 of window "Main"
```

```
    -- Delete all of the menu items from the pop-ups
    delete every menu item of menu of popup button "From"
    delete every menu item of menu of popup button "To"
    -- Add each of the unit types as menu items to both of the pop-ups
    repeat with i in my unitTypes
      make new menu item at the end of menu items of menu of
        popup button "From"
        with properties {title:i, enabled:true}
      make new menu item at the end of menu items of menu
        of popup button "To"
        with properties {title:i, enabled:true}
    end repeat
  end tell
end updateUnitTypes
```

Vous pouvez obtenir l'élément de menu courant d'un popup button avec cette instruction :

```
current menu item of popup button 1 of window 1
```

Vous pouvez régler l'élément de menu en cours d'un popup button avec cette instruction :

```
set current menu item of popup button 1 of window 1 to menu item 2 of menu of
popup button 1 of window 1
```

Vous pouvez obtenir l'index de l'élément de menu sélectionné dans un popup button avec cette instruction :

```
set currentIndex to contents of popup button 1 of window 1
```

Vous pouvez obtenir la valeur de l'élément de menu sélectionné dans un popup button avec cette instruction :

```
set currentValue to title of popup button 1 of window 1
```

Vous pouvez supprimer un élément de menu d'un popup button avec cette instruction :

```
delete menu item "Shrink It" of menu of popup button "Do Laundry" of window
"Laundromat"
```

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

Depuis la version 1.2 d'AppleScript Studio, lorsque vous faites référence à un objet menu d'un objet popup button, vous pouvez utiliser `menu item 1 of popup button 1` plutôt que l'ancienne version plus longue `menu item 1 of menu of popup button 1`, bien que cette version longue fonctionne toujours.

progress indicator

Pluriel : `progress indicators`
Hérite de : `view` (page 229)
Classe Cocoa : NSProgressIndicator

La classe progress indicator (ou barre de progression) fournit un mécanisme standard pour les "feedbacks" utilisateur. Combinée avec un objet text field (page 325), elle peut fournir à la fois une barre de progression déterminée (le temps total est connu et la barre se déplace de gauche à droite proportionnellement au pourcentage de la tâche accompli) et indéterminée (le temps total est inconnu, un cylindre rayé ou une image circulaire tourne continuellement), ainsi que des messages.

L'illustration 4.9 montre une barre de progression indéterminée.

FIG. 4.9 - *Une barre de progression indéterminée*

Vous trouverez l'objet progress indicator dans le panneau "Cocoa-Controls" d'Interface Builder. Depuis la version d'Interface Builder distribuée avec Mac OS X version 10.2, vous pouvez aussi choisir l'indicateur de progression circulaire indéterminé visible dans l'illustration 4.10. Ce type de barre de progression est souvent utilisé pour montrer une activité, comme une connexion à un réseau.

Vous pouvez régler les attributs d'un objet progress indicator dans la fenêtre Info d'Interface Builder.

FIG. 4.10 - *L'indicateur de progression circulaire indéterminé*

Propriétés des objets de la classe Progress Indicator

En plus des propriétés qu'il hérite de la classe view (page 229), un objet progress indicator possède ces propriétés :

animation delay

Accès : lecture / écriture
Classe : *integer*
Le temps d'attente entre chaque animation, par défaut cette propriété vaut 0

bezeled

Accès : lecture / écriture
Classe : *boolean*
L'objet progress indicator a-t-il un contour apparent ? Par défaut, cette propriété vaut `false`

content

Accès : lecture / écriture
Classe : *real*
La valeur de l'objet progress indicator ; synonyme de *contents*

contents

Accès : lecture / écriture
Classe : *real*
La valeur de l'objet progress indicator ; synonyme de *content*

control size

Accès : lecture / écriture
Classe : *une des constantes* de control size (page 180)
La taille de l'objet progress indicator ; par défaut, cette propriété vaut `regular size` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

control tint

Accès : lecture / écriture

Classe : *une des constantes* de control tint (page 180)
La teinte de l'objet progress indicator ; par défaut, cette propriété vaut `default tint`

indeterminate

Accès : lecture / écriture
Classe : *boolean*
La valeur de l'objet progress indicator est-elle indéterminée ? Un objet progress indicator indéterminé tourne sans interruption jusqu'à ce que le but soit atteint ; un progress indicator déterminé montre le pourcentage de la tâche effectuée ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

maximum value

Accès : lecture / écriture
Classe : *real*
La valeur maximale de l'objet progress indicator ; par défaut, cette propriété vaut 100.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

minimum value

Accès : lecture / écriture
Classe : *real*
La valeur minimale de l'objet progress indicator ; par défaut, cette propriété vaut 0.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

uses threaded animation

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'animation de l'objet progress indicator soit exécutée dans une tâche séparé ? Si l'application devient multi-tâche suite à votre volonté, ses performances pourraient devenir notablement plus lentes ; par défaut, cette propriété vaut `false`

Commandes supportées par les objets de la classe Progress Indicator

Votre script peut envoyer les commandes suivantes à un objet progress indicator :

`animate` (page 333)

`increment` (page 336)
`start` (page 340)
`stop` (page 343)

Events supportés par les objets de la classe Progress Indicator

Un objet progress indicator supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Pour un bref exemple de commandes utilisées avec un objet progress indicator, voir les commandes animate (page 333), increment (page 336),

start (page 340) et stop (page 343).

Le gestionnaire `openPanel` suivant est extrait de l'application "Mail Search" distribuée avec AppleScript Studio (avant la version 1.1 d'AppleScript Studio, cette application s'appelait "Watson"). Ce gestionnaire fait partie des multiples gestionnaires du script-objet défini dans l'application "Mail Search" pour contrôler la fenêtre "status panel".

Ce gestionnaire a un seul paramètre, `statusMessage`, lequel fournit le message affiché dans le status panel. Il est aussi relié à certaines propriétés du script, comme `initialized` et `statusPanelNibLoaded`.

Si la fenêtre "status panel" n'a pas déjà été chargée, le gestionnaire la charge, règle l'état de l'objet progress indicator de la fenêtre, indique à celui-ci de démarrer et règle le message du status panel. Il utilise alors la commande display (page 537) pour afficher le status panel (et son progress indicator attaché à une autre fenêtre).

```
on openPanel(statusMessage)
  if initialized is false then
    if not statusPanelNibLoaded then
      load nib "StatusPanel"
      set statusPanelNibLoaded to true
    end if
    tell window "status"
      set indeterminate of progress indicator "progress"
        to true
      tell progress indicator "progress" to start
        set contents of text field "statusMessage"
          to statusMessage
      end tell
      set initialized to true
    end tell
  end if
  display window "status" attached to theWindow
end openPanel
```

L'application "Mail Search" montre aussi un objet progress indicator déterminé. Regardez aussi le gestionnaire `makeStatusPanel`, lequel crée et retourne un script-objet contenant les gestionnaires chargeant un status panel contenant un objet progress indicator, règle sa valeur minimale et maximale, et lui indique d'incrémenter. L'application "Mail Search" est décrite en détails dans le guide "*Building Applications With AppleScript Studio*"

(voir la section "Autres documentations" (page 5) pour plus d'informations sur ce document).

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

La propriété *content* est apparue avec la version 1.2 d'AppleScript Studio. Vous pouvez utiliser au choix *content* et *contents*, sauf à l'intérieur d'un gestionnaire d'Events, `contents of theObject` retournant une référence à l'objet plutôt que son contenu courant. Pour obtenir dans un gestionnaire d'Events le contenu d'un objet (comme le texte contenu dans un text field (page 325)), vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

Pour un exemple montrant la différence entre *content* et *contents*, voir la section "Version" de la classe control (page 280).

Depuis la version 1.2 d'AppleScript Studio, la commande display (page 537) est préférée à la commande display panel (page 547).

Depuis la version d'Interface Builder distribuée avec Mac OS X version 10.2, vous pouvez utiliser l'objet progress indicator circulaire indéterminé visible dans l'illustration 4.10.

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" s'appelait "Watson".

`secure text field`

Pluriel : `secure text fields`
Hérite de : `text field` (page 325)
Classe Cocoa : NSSecureTextField

Cache de façon illisible le texte lors de l'affichage ou d'un autre accès via l'interface utilisateur, et est par conséquent fort utile pour un mot de passe, ou pour n'importe quel élément dans lequel une valeur sécurisée doit être saisie.

Un utilisateur peut glisser-déposer un texte dans un objet secure texte field, mais il sera affiché en accord avec les réglages courants du champ — soit avec des ronds, soit avec des caractères blancs. La valeur par défaut est d'afficher le texte sous forme de ronds. Pour plus d'informations sur la

manière de spécifier des caractères blancs, voir la section "Exemples" de la classe secure text field cell (page 314).

L'illustration 4.11 montre un champ sécurisé avec plusieurs ronds visibles. Vous pouvez créer un objet secure text field dans Interface Builder en suivant ces étapes :

1. Glissez un objet text field (page 325) du panneau "Cocoa-Text" sur la fenêtre visée.
2. Sélectionnez ce text field.
3. Dans le panneau "Custom Class" de la fenêtre Info, sélectionnez NSSecureTextField comme type de classe.

FIG. 4.11 - *Un champ sécurisé affichant des ronds*

Pour plus d'informations, voir la description des classes text field cell (page 330) et secure text field cell (page 314), ainsi que Text fields dans la documentation Cocoa.

Propriétés des objets de la classe Secure Text Field

Un objet secure text field possède uniquement les propriétés qu'il hérite de la classe text field (page 325).

Éléments des objets de la classe Secure Text Field

Un objet secure text field peut uniquement contenir les éléments qu'il hérite de la classe text field (page 325).

Events supportés par les objets de la classe Secure Text Field

Un objet secure text field supporte les gestionnaires répondant aux Events suivants :

Action

`action` (page 345)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Édition

`begin editing` (page 346)
`changed` (page 348)
`end editing` (page 350)
`should begin editing` (page 353)
`should end editing` (page 354)

Clavier

`keyboard up` (page 134)

Souris

`mouse entered` (page 139)
`mouse exited` (page 139)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Un objet secure text field n'a pas de propriétés ou d'éléments propres scriptables. Toutefois, il hérite l'ensemble de ses propriétés et de ses éléments de la classe text field (page 325). Voir aussi la classe secure text field cell (page 314).

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

secure text field cell

Pluriel : `secure text field cells`
Hérite de : `text field cell` (page 330)
Classe Cocoa : NSSecureTextFieldCell

Fonctionne avec un objet secure text field (page 311) afin de fournir un champ texte dont l'affichage de la valeur est masquée à l'utilisateur.

Fournit son propre éditeur de champ, lequel n'affiche pas le texte ou interdit à l'utilisateur de couper, copier et coller sa valeur. Toutefois, vous pouvez glisser-déposer un texte dans un champ sécurisé.

Pour plus d'informations, voir la classe secure text field (page 311), ainsi que Text Fields dans la documentation Cocoa.

Vous pouvez créer et accéder à un objet secure text field cell dans Interface Builder en suivant ces étapes :

1. Utilisez les étapes décrites dans la classe secure text field (page 311) pour créer le champ sécurisé.
2. Sélectionnez l'objet secure text field.
3. Tout en maintenant appuyée la touche Option, saisissez avec la souris une des poignées de redimensionnement. En même temps que vous glissez, Interface Builder crée un objet matrix (page 289) contenant plusieurs objets text field cell (page 330).
4. Cliquer une fois sélectionne l'objet matrix ; double-cliquer sélectionne un objet text field cell de l'objet matrix. Pour chaque cellule :

   (a) sélectionnez la cellule

(b) ouvrez la fenêtre Info et utilisez le menu déroulant pour afficher le panneau “Custom Class”

(c) modifiez la classe de la cellule en NSSecureTextFieldCell

Propriétés des objets de la classe Secure Text Field Cell

En plus des propriétés qu'il hérite de la classe text field cell (page 330), un objet secure text field cell possède ces propriétés :

echos bullets
Accès : lecture / écriture
Classe : *boolean*
Faut-il que les caractères soient masqués par des ronds ? Par défaut, cette propriété vaut `true` ; si elle est réglée sur `false`, le masquage se fait avec des caractères blancs

Éléments des objets de la classe Secure Text Field Cell

Un objet secure text field cell peut uniquement contenir les éléments qu'il hérite de la classe text field cell (page 330).

Exemples

L'instruction suivante modifie la propriété *echos bullets* d'un objet secure text field cell. Par défaut, les caractères sont masqués par des ronds. Lorsque les ronds sont désactivés, les caractères sont masqués par des caractères blancs.

```
set echos bullets of cell of secure text field 1 of window 1 to false
```

slider

Pluriel : `sliders`
Hérite de : `control` (page 280)
Classe Cocoa : NSSlider

Affiche une rangée de valeurs et comporte un indicateur, ou poignée, lequel indique le réglage courant.

Un objet slider peut facultativement comporter une graduation. L'utilisateur déplacera la poignée le long de la barre de l'objet slider pour modifier le réglage.

L'objet slider détermine automatiquement s'il doit être horizontal ou vertical grâce à sa taille. Si l'objet slider est plus large qu'il est grand, il sera horizontal. Dans les autres cas, il sera vertical. Un slider vertical a sa valeur minimale en bas ; un slider horizontal l'a à sa gauche.

L'illustration 4.12 montre des objets slider verticaux et horizontaux avec différentes sortes de poignées, et avec et sans graduation. Vous trouverez ces types d'objet dans le panneau "Cocoa-Controls" d'Interface Builder.

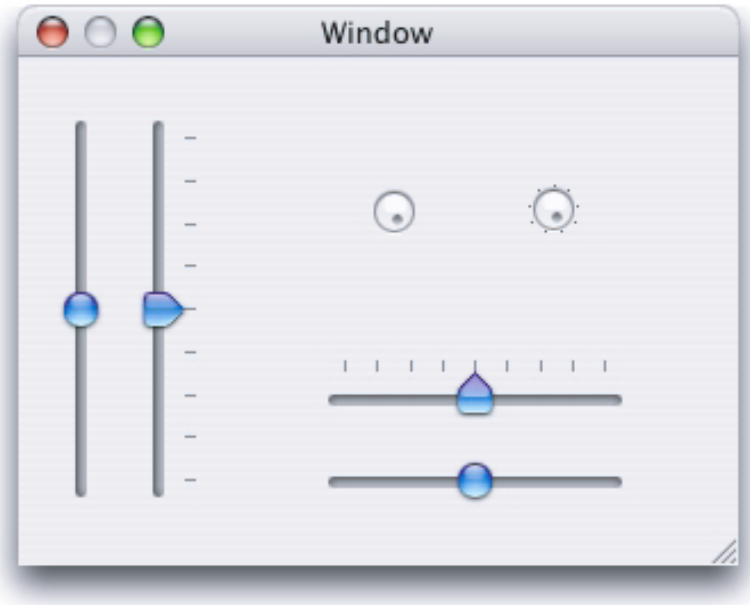

FIG. 4.12 - *Sliders horizontaux et verticaux, sans et avec graduation*

Pour plus d'informations, voir Sliders dans la documentation Cocoa.

Propriétés des objets de la classe Slider

En plus des propriétés qu'il hérite de la classe control (page 280), un objet slider possède ces propriétés :

alternate increment value

Accès : lecture / écriture
Classe : *real*
La valeur d'incrémentation alternative est la quantité avec laquelle

l'objet slider modifiera sa valeur lorsque l'utilisateur fera glisser la poignée avec la touche Option enfoncée (ou touche Alt sur certains claviers)

image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image de l'objet slider ; le réglage de l'image d'un slider dans Cocoa est dépréciée, par conséquent vous ne devrez pas régler l'image dans un script

knob thickness

Accès : lecture / écriture
Classe : *real*
L'épaisseur de la poignée

maximum value

Accès : lecture / écriture
Classe : *real*
La valeur maximale de l'objet slider ; par défaut, cette propriété vaut 100.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

minimum value

Accès : lecture / écriture
Classe : *real*
La valeur minimale de l'objet slider ; par défaut, cette propriété vaut 0.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

number of tick marks

Accès : lecture / écriture
Classe : *integer*
Le nombre de marques de l'objet slider ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder ; certains sliders n'ont pas de graduation ; pour les autres, le nombre par défaut est 11

only tick mark values

Accès : lecture / écriture
Classe : *boolean*
Faut-il que seules les valeurs autorisées correspondent aux marques ? Vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder ; pour les sliders ne comportant pas de graduation, la valeur par

défaut est `false` ; pour les sliders avec, la valeur par défaut est `true`

tick mark position

Accès : lecture / écriture
Classe : *une des constantes* de tick mark position (page 190)
La position des marques (au-dessus ou au-dessous de l'objet slider horizontal ou à gauche ou à droite d'un slider vertical) ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de l'objet slider ; le réglage du titre d'un objet slider dans Cocoa est dépréciée, par conséquent vous ne devrez pas régler le titre dans un script

title cell

Accès : lecture / écriture
Classe : *text field cell* (page 330)
La cellule du titre ; comme la propriété *title*, vous ne devrez pas utiliser cette propriété dans vos scripts

title color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du titre ; comme la propriété *title*, vous ne devrez pas utiliser cette propriété dans vos scripts

title font

Accès : lecture / écriture
Classe : *font* (page 54)
La police du titre ; comme la propriété *title*, vous ne devrez pas utiliser cette propriété dans vos scripts

vertical

Accès : lecture / écriture
Classe : *boolean*
Le slider est-il orienté verticalement ? Vous pouvez choisir des sliders horizontaux ou verticaux dans Interface Builder

Éléments des objets de la classe Slider

Un objet slider peut uniquement contenir les éléments qu'il hérite de la classe control (page 280).

Events supportés par les objets de la classe Slider

Un objet slider supporte les gestionnaires répondant aux Events suivants :

Action

`action` (page 345)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Lorsque vous ajoutez un objet slider à une fenêtre dans Interface Builder, vous pouvez régler divers attributs de cet objet, comme ses valeurs minimales, maximales et courantes (ou de départ), et son nombre de marques. Le gestionnaire action (page 345) suivant est extrait de l'application "Language Translator" distribuée avec AppleScript Studio. Pour cette application, le slider est réglé pour autoriser une graduation de 0.0 à 7.0, laquelle correspond à la graduation du volume sonore réglable avec le complément standard `set volume` (où 0.0 correspond au silence et 7.0 au volume à fond).

Le gestionnaire action (page 345), lequel est appelé lorsqu'un utilisateur modifie le réglage du slider, obtient le slider de la fenêtre depuis le paramètre `theObject` transmis. Il obtient alors une valeur de volume basée sur le réglage courant du slider et appelle `set volume` pour ajuster le volume auquel le texte traduit sera lu.

```
on action theObject
  set enabled of slider of window "Language Translator" to true
  set volumevalue to contents of slider "volumeslider"
    of window "Language Translator" as integer
  set volume volumevalue
end action
```

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

stepper

Pluriel : `steppers`
Hérite de : `control` (page 280)
Classe Cocoa : NSStepper

Un contrôle composé de deux petites flèches pouvant incrémenter ou décrémenter une valeur apparaissant à côté, comme une date ou une heure.

L'illustration 4.13 montre un objet stepper à droite d'un champ texte affichant la valeur de cet objet.

FIG. 4.13 - *Un objet stepper*

Vous trouverez l'objet stepper dans le panneau "Cocoa-Controls" d'Interface Builder. Vous pouvez régler plusieurs attributs des objets stepper dans la fenêtre Info d'Interface Builder.

Pour plus d'informations, voir Steppers dans la documentation Cocoa.

Propriétés des objets de la classe Stepper

En plus des propriétés qu'il hérite de la classe control (page 280), un objet stepper possède ces propriétés :

auto repeat

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet stepper s'auto-incrémente ou décrémente lorsqu'il est maintenu cliqué ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

increment value

Accès : lecture / écriture
Classe : *real*
La quantité d'incrémentation de l'objet stepper ; par défaut, cette propriété vaut 1.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

maximum value

Accès : lecture / écriture
Classe : *real*
La valeur maximale de l'objet stepper ; par défaut, cette propriété vaut 59.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

minimum value

Accès : lecture / écriture
Classe : *real*
La valeur minimale de l'objet stepper ; par défaut, cette propriété vaut 0.0 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

value wraps

Accès : lecture / écriture
Classe : *boolean*
N'ayant pas réussi à traduire l'explication, je laisse le texte original, si quelqu'un arrive à trouver une traduction claire, merci de me la faire suivre.
Does the value of the stepper wrap ? if true, then when incrementing or decrementing, the value will wrap around to the minimum or maximum value ; if false, the value will stay pinned at the minimum or maximum ; default is true ; you can set this property in the Info window in Interface Builder

Éléments des objets de la classe Stepper

Un objet stepper peut uniquement contenir les éléments qu'il hérite de la classe control (page 280).

Events supportés par les objets de la classe Stepper

Un objet stepper supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

Lorsque vous ajoutez un objet stepper à une fenêtre dans Interface Builder, vous pouvez régler divers attributs de cet objet, comme ses valeurs minimales, maximales et courantes (ou de départ), et sa valeur d'incrémentation. Le gestionnaire clicked (page 348) suivant est extrait de l'application "Drawer" distribuée avec AppleScript Studio. Dans cette application, l'objet stepper contrôlant la propriété *leading offset* du tiroir est réglé pour autoriser une distance de 0.0 à 1000.0, avec une quantité d'incrémentation de 1.0.

Ce gestionnaire clicked, lequel est appelé lorsque l'utilisateur clique sur le stepper, vérifie le paramètre `theObject` pour déterminer quel objet l'a appelé. S'il s'agit de l'objet stepper, il obtient sa valeur et règle la propriété *leading offset* de l'objet drawer. Finalement, il règle la valeur affichée dans le champ texte associé (notez qu'il n'a pas besoin de convertir la valeur en chaîne de caractères pour régler le contenu du champ texte).

```
on clicked theObject
  tell window "main"
```

```
    if theObject is equal to button "drawer" then
  ...
    else if theObject is equal to stepper "leading offset" then
      set theValue to (contents of stepper "leading offset")
        as integer
      set leading offset of drawer "drawer" to theValue
      set contents of text field "leading offset" to theValue
  ...
```

Les instructions suivantes, extraites du gestionnaire action (page 345) de l'application "Drawer", montre comment régler la valeur de l'objet stepper à partir de la valeur saisie dans le champ texte associé.

```
on action theObject
  set textValue to contents of theObject
  ...
  if theObject is equal to text field "leading offset" then
    set leading offset of drawer "drawer" to textValue
    set contents of stepper "leading offset" to textValue
...
```

Pour régler la valeur d'un objet stepper, et afficher la même valeur dans un champ texte, vous pouvez utiliser les instructions suivantes :

```
tell window "main"
  set contents of stepper "stepperName" to myNumber
  set contents of text field "textFieldName" to myNumber
end tell
```

Pour obtenir la valeur, utilisez cette instruction, laquelle retourne un nombre réel :

```
tell window "main"
  set currentStepperValue to contents of stepper "stepperName"
end tell
```

Version

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d'AppleScript Studio.

text field

Pluriel : `text fields`
Hérite de : `control` (page 280)
Classe Cocoa : NSTextField

Fournit la capacité de saisir, d'afficher et d'éditer du texte, ainsi que d'afficher du texte non-éditable pouvant être utilisé pour les étiquettes.

Vous trouverez l'objet text field dans le panneau "Cocoa-Text" d'Interface Builder. Vous pouvez régler plusieurs attributs des objets text field dans la fenêtre Info d'Interface Builder. Pour plus d'informations sur le réglage de la police de caractères, la couleur et d'autres attributs d'un objet text field, voir la section "Exemples" de la classe font (page 54).

Vous pouvez connecter un gestionnaire action (page 345) à un objet text field dans Interface Builder pour récupérer le contrôle lorsqu'un utilisateur a fini l'édition (soit en tabulant dans un autre champ ou en appuyant sur la touche Entrée). Dans Interface Builder, vous pouvez aussi facultativement régler l'objet text field pour qu'il appelle uniquement son gestionnaire action lorsque la touche Entrée est appuyée (et non lorsque l'utilisateur tabule vers un autre champ).

Les applications AppleScript Studio supporte automatiquement la tabulation entre champs ajoutés à une fenêtre. Par défaut, l'ordre de tabulation va de gauche à droite puis en-dessous, et est indépendant de l'ordre dans lequel vous insérez les objets text field. Voir "Enabling Tabbing Between Objects" dans l'aide d'Interface Builder pour une description sur la manière de régler l'ordre de la tabulation entre des objets text fields ou d'autres objets ; vous pouvez spécifier le premier objet recevant les Events clavier (ou le first responder initial), et chaque objet successif de la chaîne responder.

L'illustration 4.14 montre un text field étiquette ("System Font Text") et un champ de saisie. Pour plus d'informations, voir Text Fields dans la documentation Cocoa.

Propriétés des objets de la classe Text Field

En plus des propriétés qu'il hérite de la classe control (page 280), un objet text field possède ces propriétés :

allows editing text attributes

Accès : lecture / écriture
Classe : *boolean*

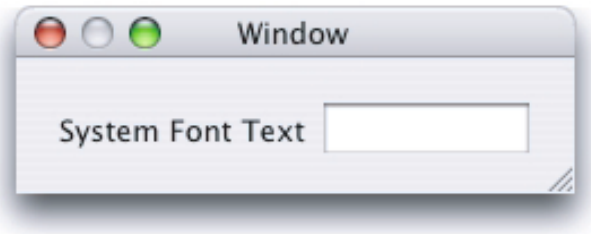

FIG. 4.14 - *Des objets text field utilisés comme étiquette et champ de saisie*

L'utilisateur peut-il éditer les attributs de polices de l'objet text field ? Par défaut, cette propriété vaut `false`

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond de l'objet text field ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par défaut, cette propriété vaut {65535, 65535, 65535}, ou la couleur blanche ; vous pouvez la régler dans la fenêtre Info d'Interface Builder ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0}

bezeled

Accès : lecture / écriture
Classe : *boolean*
L'objet text field a-t-il un contour apparent ? Par défaut, cette propriété vaut `true`

bordered

Accès : lecture / écriture
Classe : *boolean*
L'objet text field a-t-il une bordure ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet text field dessine son fond derrière le texte ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder, bien que vous devrez d'abord régler le type

de bordure sur `no border`

editable

Accès : lecture / écriture
Classe : *boolean*
L'objet text field est-il éditable ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

imports graphics

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet text field supporte le glisser-déposer d'images ? Par défaut, cette propriété vaut `false`

next text

Accès : lecture / écriture
Classe : *text field* (page 325)
Non supportée dans la version 1.3 d'AppleScript Studio ; se servir de la méthode de la classe NSMatrix sur laquelle cette propriété est basée n'est pas encouragée, aussi cette propriété risque de ne jamais être supportée ; le prochain éditeur de l'objet text field

previous text

Accès : lecture / écriture
Classe : *text field* (page 325)
Non supportée dans la version 1.3 d'AppleScript Studio ; se servir de la méthode de la classe NSMatrix sur laquelle cette propriété est basée n'est pas encouragée, aussi cette propriété risque de ne jamais être supportée ; le précédent éditeur de l'objet text field

selectable

Accès : lecture / écriture
Classe : *boolean*
Le contenu de l'objet text field peut-il être sélectionné ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

text color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du texte ; une liste de trois nombres entiers contenant les

valeurs de chaque composant de la couleur ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0} ; par défaut, cette propriété vaut {0, 0, 0} ou la couleur noire ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

Éléments des objets de la classe Text Field

Un objet text field peut uniquement contenir les éléments qu'il hérite de la classe control (page 280).

Events supportés par les objets de la classe Text Field

Un objet text field supporte les gestionnaires répondant aux Events suivants :

Action

`action` (page 345)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Édition

`begin editing` (page 346)
`changed` (page 348)
`end editing` (page 350)
`should begin editing` (page 353)
`should end editing` (page 354)

Clavier

`keyboard up` (page 134)

Souris

`mouse entered` (page 139)

`mouse exited` (page 139)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

La plupart des applications distribuées avec AppleScript Studio montrent comment travailler avec les objets text field. Par exemple, l'application "Currency Converter", disponible depuis la version 1.1 d'AppleScript Studio, utilise l'instruction suivante pour obtenir le texte de l'objet text field. Dans cette instruction, le texte est stocké dans la variable `theRate`, où `"rate"` spécifie le nom AppleScript de l'objet text field. Vous indiquerez le nom AppleScript dans le champ "Name" du panneau AppleScript de la fenêtre Info d'Interface Builder.

```
set theRate to contents of text field "rate"
```

Cette même application utilise l'instruction suivante pour régler le texte dans un autre text field afin d'afficher la conversion monétaire :

```
set contents of text field "total" to theRate * theAmount
```

Notez que l'objet text field affiche automatiquement le résultat sous forme de texte.

Vous pouvez utiliser l'instruction suivante pour régler la couleur du texte d'un champ texte sur le rouge :

```
set the text color of text field "text1" of window 1 to {65535, 0, 0}
```

Version

Le support des Events de Glisser-Déposer est apparu dans la version 1.2 d'AppleScript Studio. Voir Drag and Drop Suite pour plus de détails. En particulier, la description du gestionnaire conclude drop (page 481) fournit

des informations sur le support du Glisser-Déposer des objets text view (page 569) et text field (page 325).

Les propriétés *next text* et *previous text* de cette classe ne sont pas supportées dans la version 1.3 d'AppleScript Studio. Le support de ces propriétés risque de ne jamais être ajouté.

text field cell

Pluriel : `text field cells`
Hérite de : `cell` (page 264)
Classe Cocoa : NSTextFieldCell

Ajoute au texte affiché les capacités d'un objet cell en fournissant la possibilité de régler en même temps la couleur du texte et de son fond, ainsi que de spécifier si la cellule dessine son fond ou non.

Vous pouvez créer et accéder dans Interface Builder à un objet text field cell en suivant ces étapes :

1. Glissez un objet text field du panneau "Cocoa-Text" sur la fenêtre visée.
2. Sélectionnez l'objet text field.
3. Maintenez enfoncée la touche Option et avec le curseur de la souris, étirez une des poignées de redimensionnement. En même temps que vous faites glisser la souris, Interface Builder crée un objet matrix (page 289) contenant plusieurs objet text field cell. Suivant si vous glissez à l'horizontal ou à la verticale, les objets text field cell seront respectivement alignés à l'horizontal ou à la verticale.
4. Cliquer une seule fois sélectionnera l'objet matrix ; double-cliquer sélectionnera un des objets text field cell de l'objet matrix.

Propriétés des objets de la classe Text Field Cell

En plus des propriétés qu'il hérite de la classe cell (page 264), un objet text field cell possède ces propriétés :

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond de la cellule ; une liste de trois nombres entiers

contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0} ; par défaut, cette propriété vaut {65535, 65535, 65535}, ou la couleur blanche ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la cellule dessine son fond ? Vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

text color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du texte ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0} ; par défaut, cette propriété vaut {0, 0, 0}, ou la couleur noire ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

Events supportés par les objets de la classe Text Field Cell

Un objet text field cell supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Nib

`awake from nib` (page 123)

Exemples

Vous ne scripterez généralement pas un objet text field cell. Vous pouvez à la place scripter les propriétés identiques d'un objet text field (page 325) (*background color*, *draws background* et *text color*). Toutefois, si vous avez besoin d'accéder aux propriétés d'un objet text field cell, vous pouvez le faire avec les instructions suivantes :

```
set textFieldCell to cell 1 of matrix 1
set myColor to background color of textFieldCell
```

```
set draws background of textFieldCell to true
```

Chapitre 2

Commandes

Les objets basés sur les classes de la suite Control View supportent les commandes suivantes. Une **commande** est un mot ou une phrase qu'un script peut utiliser pour demander une action. Pour déterminer les commandes supportées par chaque classe, voir les descriptions propres à chacune.

animate 333
go 334
highlight 335
increment 336
pause 337
perform action 337
play 338
resume 339
scroll 339
start 340
step back 341
step forward 342
stop 343
synchronize 343

`animate`

Fait progresser l'animation d'un objet progress indicator indéterminé par étapes.

Syntaxe

`animate` *reference* obligatoire

Paramètres

reference
La référence de l'objet progress indicator (page 306) à animer

Exemples

Étant donné un objet window (page 76) nommé "main" comportant un objet progress indicator (page 306) nommé "barber pole", l'instruction suivante provoque la progression du progress indicator en une seule étape. Pour animer de façon continue la barre de progression, vous pouvez, soit utiliser cette instruction de façon répétée dans une boucle, soit utiliser la commande start (page 340).

```
tell progress indicator "barber pole" of window "main" to animate
```

L'instruction suivante est équivalente :

```
animate progress indicator "barber pole" of window "main"
```

go

Saute à l'emplacement spécifié dans le film (l'objet direct).

Pour des informations de même nature, voir la classe movie view (page 296).

Syntaxe

`go`	*reference*	obligatoire
`[to]`	*une constante*	facultatif

Paramètres

reference
La référence du film dans lequel le saut doit être réalisé

`[to]` *une des constantes* de go to (page 183)
L'emplacement dans le film où doit se placer la tête de lecture

Exemples

Le gestionnaire choose menu item (page 505) suivant est extrait de l'application "Talking Head" distribuée avec AppleScript Studio. Ce gestionnaire utilise la propriété *tag* de l'objet menu item (page 499) du menu "Movie" pour déterminer quelle commande exécuter.

Pour l'élément de menu "Go To Beginning", le gestionnaire utilise la commande Go To pour placer la tête de lecture au début du film. Vous pouvez régler la valeur `tag` d'un élément de menu dans le panneau "Attributes" de la fenêtre Info d'Interface Builder. Les valeurs `tag` de cet exemple sont extraites de l'application "Talking Head".

```
on choose menu item theMenuItem
  tell window "main"
    set theCommand to tag of theMenuItem
    if theCommand is equal to 1001 then
  ...
    else if theCommand is equal to 1006 then
      tell movie view "movie" to go to beginning frame
  ...
```

highlight

Non supportée dans la version 1.3 d'AppleScript Studio. N'utilisez pas cette commande.

Syntaxe

`highlight` *reference* obligatoire

Paramètres

reference

La référence de l'objet à illuminer

increment

Incrémente l'objet spécifié avec la quantité spécifiée, ou avec 1 si aucune quantité n'est spécifiée.

Syntaxe

`increment`	*reference*	obligatoire
`[by]`	*real*	facultatif

Paramètres

reference
La référence de l'objet à incrémenter

`[by]` *real*
La quantité à ajouter

Exemples

Les instructions suivantes sont extraites du gestionnaire `incrementPanel` de l'application "Mail Search" distribuée avec AppleScript Studio. Ce gestionnaire est un des multiples gestionnaires faisant partie du script-objet défini dans cette application pour contrôler le "status panel".

```
...
  tell window "status"
    tell progress indicator "progress" to increment by 1
    ...
    end tell
...
```

Ces instructions visent l'objet progress indicator (page 306) déterminé d'une fenêtre et utilise la commande Increment pour l'incrémenter.

Version

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" se nommait "Watson".

pause

Non supportée dans la version 1.1 d'AppleScript Studio. Met sur pause l'activité courante.

Syntaxe

`pause` *reference* obligatoire

Paramètres

reference
La référence de l'objet à mettre sur pause

perform action

Indique à l'objet receveur d'exécuter son action (provoque l'exécution du gestionnaire concerné).

Par exemple, vous pouvez dire à un objet d'interface, comme un objet button (page 254), d'exécuter son gestionnaire clicked (page 348), fournissant par conséquent une manière directe de scripter l'interface utilisateur. Notez, toutefois, qu'appeler le gestionnaire clicked ne produira pas le retour visuel qu'un utilisateur verrait s'il avait réellement cliqué sur le bouton. Pour d'autres limitations du scripting de l'interface utilisateur, voir le guide "*Building Applications With AppleScript Studio*", disponible dans l'aide de Project Builder ou sur le site d'Apple.

Syntaxe

`perform action` *reference* obligatoire

Paramètres

reference
La référence de l'objet devant exécuter son action

Exemples

Vous pouvez utiliser les instructions suivantes dans l'application Éditeur de Scripts pour provoquer l'exécution du gestionnaire `clicked` du bouton "Drawer" de l'application "Drawer" distribuée avec AppleScript Studio, ce

gestionnaire soit ouvrira le tiroir, soit le fermera, en fonction de son état courant. Ces mêmes instructions fonctionneront dans le script d'une application AppleScript Studio (bien que vous n'aurez pas besoin de les encadrer par un bloc Tell).

```
tell application "Drawer"
  set theButton to button "Drawer" of window "main"
  tell theButton to perform action
end tell
```

Discussion

La commande Perform Action généralement ne fait rien tant que l'objet spécifié n'a pas de gestionnaire Action — un gestionnaire comme clicked (page 348) ou double clicked (page 349) dans le groupe "Action" de la fenêtre Info de l'objet dans Interface Builder. Vous pouvez, toutefois, utiliser la commande Perform Action avec les éléments de menu, en utilisant une syntaxe telle que celle-ci :

```
tell menu item 1 of menu 1 of main menu to perform action
```

play

Joue l'objet. La commande Play est supportée par la classe movie view (page 296).

Syntaxe

`play` *reference* obligatoire

Paramètres

reference
La référence du film à jouer

Exemples

Vous pouvez indiquer à un objet movie view (page 296) de jouer avec les instructions suivantes :

```
tell window "main"
```

```
  tell movie view "movie" to play
end tell
```

Pour un exemple plus complexe, voir l'application "Talking Head" distribuée avec AppleScript Studio.

resume

Non supportée dans la version 1.3 d'AppleScript Studio. Reprend l'activité précédente.

Syntaxe

`resume`	*reference*	obligatoire

Paramètres

reference
: La référence de l'objet qui doit reprendre son activité précédente

scroll

Fait défiler l'objet spécifié.

Non supportée dans la version 1.3 d'AppleScript Studio. Toutefois, voir la section "Exemples", ci-dessous, pour plus d'informations sur la manière de faire défiler le texte d'un objet text view (page 569).

Syntaxe

`scroll`	*reference*	obligatoire
`[item at index]`	*integer*	facultatif
`[to]`	*une constante*	facultatif

Paramètres

reference
: La référence de l'objet devant être déroulé

`[item at index]` *integer*
: L'index de l'élément jusqu'où il faut faire défiler

`[to]` *une des constantes* de scroll to location (page 186)
L'emplacement jusqu'où doit se faire le défilement

Exemples

Bien que la commande Scroll ne soit pas supportée dans la version 1.3 d'AppleScript Studio, vous pouvez utiliser à la place la commande call method (page 94) pour faire défiler. Pour un objet text view (page 569), par exemple, vous pouvez utiliser la méthode `scrollRangeToVisible:` de la classe NSText (la classe text view hérite de la classe text (page 569), comme NSTextView hérite de NSText).

Pour un objet window (page 76) nommé "main" comportant un objet text view (page 569) nommé "myText" dans un objet scroll view (page 213) nommé "scroller", les instructions suivantes feront défiler jusqu'au bas de l'objet text view.

```
tell text view "myText" of scroll view "scroller" of window "main"
  set theText to contents
  set theLength to (length of theText)
  call method "scrollRangeToVisible:" of object it
    with parameter {theLength, theLength}
end tell
```

Substituer l'instruction suivante à la commande Call Method du script précédent fera défiler jusqu'en haut de l'objet text view :

```
call method "scrollRangeToVisible:" of object it with parameter {0, 0}
```

start

Démarre un objet.

Diverses classes supportent cette commande. Par exemple, vous pouvez l'utiliser pour démarrer l'animation d'un objet progress indicator (page 306) indéterminé, ce qui met en route la vrille. La commande Start ne fait rien avec un objet progress indicator déterminé.

Vous pouvez aussi utiliser la commande Start pour jouer un film dans un objet movie view (page 296).

Syntaxe

`start` *reference* obligatoire

Paramètres

reference
: La référence de l'objet à démarrer

Exemples

Étant donné un objet window (page 76) nommé "main" comportant un objet progress indicator (page 306) indéterminé nommé "barber pole", l'instruction suivante provoquera le démarrage de l'animation de l'objet progress indicator.

```
tell progress indicator "barber pole" of window "main" to start
```

L'instruction suivante est équivalente :

```
start progress indicator "barber pole" of window "main"
```

Pour un exemple plus complexe, voir l'application "Talking Head" distribuée avec AppleScript Studio.

`step back`

Repositionne la lecture du film sur la frame précédant immédiatement la frame courante. Si le film est joué, alors il s'arrêtera sur la nouvelle frame.

Syntaxe

`step back` *reference* obligatoire

Paramètres

reference
: La référence de l'objet movie view (page 296) recevant la commande Step Back

Exemples

Vous pouvez indiquer à un objet movie view (page 296) de revenir en arrière avec les instructions suivantes :

```
tell window "main"
  tell movie view "movie" to step back
end tell
```

Pour un exemple plus complexe, voir l'application "Talking Head" distribuée avec AppleScript Studio.

step forward

Repositionne la lecture du film sur la frame suivant immédiatement la frame courante. Si le film est joué, alors il s'arrêtera sur la nouvelle frame.

Syntaxe

`step forward` *reference* obligatoire

Paramètres

reference

La référence de l'objet movie view (page 296) recevant la commande Step Forward

Exemples

Vous pouvez indiquer à un objet movie view (page 296) d'avancer avec les instructions suivantes :

```
tell window "main"
  tell movie view "movie" to step forward
end tell
```

Pour un exemple plus complexe, voir l'application "Talking Head" distribuée avec AppleScript Studio.

stop

Arrête un objet.

Diverses classes supportent cette commande. Par exemple, vous pouvez l'utiliser pour arrêter l'animation d'un objet progress indicator (page 306) indéterminé, ce qui arrête la vrille. La commande Stop ne fait rien avec un objet progress indicator déterminé.

Vous pouvez aussi utiliser la commande Stop pour arrêter un film dans un objet movie view (page 296).

Syntaxe

`stop` *reference* obligatoire

Paramètres

reference
La référence de l'objet à arrêter

Exemples

Étant donné un objet window (page 76) nommé "main" comportant un objet progress indicator (page 306) indéterminé nommé "barber pole", l'instruction suivante provoquera l'arrêt de l'animation de l'objet progress indicator.

```
tell progress indicator "barber pole" of window "main" to stop
```

Vous pouvez indiquer à un objet movie view (page 296) de s'arrêter avec les instructions suivantes :

```
tell window "main"
  tell movie view "movie" to stop
end tell
```

Pour un exemple plus complexe, voir l'application "Talking Head" distribuée avec AppleScript Studio.

synchronize

Enregistre sur le disque toute modification touchant les valeurs utilisateur par défaut et met à jour les valeurs non-modifiées avec celle sur le

disque.

Non supportée dans la version 1.3 d'AppleScript Studio, mais voir la section "Exemples" pour une solution de remplacement.

Syntaxe

`synchronize` *reference* obligatoire

Paramètres

reference

La référence de l'objet user-defaults (page 72) devant être synchronisé

Résultats

boolean

Retournera `false` si elle ne peut pas enregistrer de données sur le disque ; `true` dans l'autre cas

Exemples

Vous pouvez utiliser la commande call method (page 94) pour appeler la méthode `synchronize`, comme dans ce qui suit. Ces appels retourneront `false` si les données ne peuvent pas être enregistrées sur le disque, autrement `true`.

```
-- Fonctionne dans Jaguar ainsi que dans les versions précédentes
set succeeded to call method "synchronize" of object user defaults

-- Fonctionne uniquement dans Jaguar
set succeeded to call method "synchronize" of user defaults
```

Chapitre 3

Events

Les objets basés sur les classes de la suite Control View supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

action 345
begin editing 346
changed 348
clicked 348
double clicked 349
end editing 350
selection changed 351
selection changing 352
should begin editing 353
should end editing 354
will dismiss 355
will pop up 356

`action`

Appelé lorsqu'une action survient pour un objet. Ce gestionnaire doit son nom au concept Cocoa de l' action — une méthode pouvant être déclenchée par les objets de l'interface utilisateur. Dans AppleScript Studio, les gestionnaires action sont clicked (page 348), double clicked (page 349) et Action.

Le gestionnaire Action lui-même est supporté par les classes comme combo box (page 274), popup button (page 301), secure text field (page 311), slider (page 315), stepper (page 320), text field (page 325) et text field cell (page 330). Par exemple, le gestionnaire Action d'un objet text field est généralement déclenché lorsque l'utilisateur essaie de tabuler hors du champ ou appuie sur la touche Retour. Vous pouvez régler le bouton radio "Send Action" d'un objet text field dans le panneau "Attributes" de la fenêtre Info d'Interface Builder.

Vous pouvez utiliser la commande perform action (page 337) pour provoquer l'appel d'un gestionnaire Action.

Syntaxe

`action` *reference* obligatoire

Paramètres

reference

La référence de l'objet ayant reçu l'action

Exemples

Le gestionnaire Action suivant est extrait du fichier `Application.applescript` de l'application "Unit Converter" distribuée avec AppleScript Studio. Ce gestionnaire vérifie juste si l'objet pour lequel le gestionnaire a été appelé est un objet text field (page 325) particulier. Si oui, il appelle un autre gestionnaire pour exécuter la conversion (comme convertir des yards en mètres).

```
on action theObject
  if theObject is equal to text field "Value" of box 1 of window "Main" then
    my convert()
  end if
end action
```

begin editing

Appelé avant que l'édition ne commence.

Vous utiliserez généralement ce gestionnaire avec les objets text field (page 325), text view (page 569) ou autres. Le gestionnaire ne peut pas annuler l'opération d'édition, mais peut la préparer.

Par exemple, lorsque l'utilisateur appuiera pour la première fois sur une touche du clavier pour saisir un caractère dans un objet text field (page 325), AppleScript Studio appellera le gestionnaire should begin editing (page 353) (s'il est installé), lequel pourra refuser d'autoriser l'édition. Puis, si l'édition est autorisée, le gestionnaire Begin Editing (s'il est installé). Le caractère ne sera pas saisi dans le champ tant que l'application ne retournera pas le résultat du gestionnaire Begin Editing.

Pour continuer avec cet exemple, AppleScript Studio appellera le gestionnaire changed (page 348) (s'il est installé) après que le caractère soit saisi. Lorsque l'utilisateur appuiera sur la touche Tabulation ou essaiera de compléter l'édition dans ce champ, AppleScript Studio appellera le gestionnaire should end editing (page 354), lequel pourra refuser d'autoriser la fin de l'édition (si, par exemple, le champ contient une saisie incorrecte). Puis, si la fin de l'édition est autorisée, AppleScript Studio appellera le gestionnaire end editing (page 350) (s'il est installé), où l'application pourra exécuter toute instruction pour que l'édition soit complète.

Syntaxe

`begin editing` *reference* obligatoire

Paramètres

reference

La référence des objets text field (page 325), text view (page 569) ou d'autres objets de même nature pour lesquels l'édition commencera

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Begin Editing à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer l'édition.

```
on begin editing theObject
  (* Perform operations here before editing. *)
end begin editing
```

changed

Appelé après que le contenu d'un objet ait été modifié.

Généralement utilisé pour indiquer que l'édition a provoqué une modification du texte d'un objet text field (page 325), text view (page 569) ou autres. Lorsque ce gestionnaire est appelé, il est trop tard pour faire la validation — utiliser plutôt le gestionnaire should end editing (page 354) à la place. Voir aussi begin editing (page 346), should begin editing (page 353) et end editing (page 350).

Syntaxe

`changed` *reference* obligatoire

Paramètres

reference

La référence de l'objet dans lequel l'édition a provoqué une modification

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Changed à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour répondre à toute modification provoquée par l'édition.

```
on changed theObject
  (* Perform operations here after object changed due to editing. *)
end changed
```

clicked

Appelé lorsque l'utilisateur clique sur l'objet.

Presque toutes les sous-classes de la classe control (page 280) supportent le gestionnaire Clicked.

Syntaxe

`clicked` *reference* obligatoire

Paramètres

reference
La référence de l'objet ayant été cliqué

Exemples

Le gestionnaire Clicked suivant est pour le bouton "Convert" de l'application "Currency Converter" distribuée avec AppleScript Studio. Cette application fournit des champs texte pour la saisie de la somme et du taux de conversion, et un champ pour l'affichage du résultat, ainsi que le bouton "Convert" pour lancer la conversion.

```
on clicked theObject
  tell window of theObject
    try
      set theRate to contents of text field "rate"
      set theAmount to contents of text field "amount" as number
      set contents of text field "total" to theRate * theAmount
    on error
      set contents of text field "total" to 0
    end try
  end tell
end clicked
```

Seul un objet (le bouton "Convert") de l'application "Currency Converter" supporte le gestionnaire Clicked, aussi, dans ce gestionnaire, l'application sait que le paramètre `theObject` référence le bouton. Le gestionnaire obtient la fenêtre de cet objet ainsi il peut accéder aux champs texte de cette même fenêtre.

double clicked

Appelé après que l'utilisateur double-clique sur l'objet.

Les sous-classes de control (page 280) comme outline view (page 392) et table view (page 403) supportent le gestionnaire Double Clicked.

Syntaxe

`double clicked` *reference* obligatoire

Paramètres

reference

La référence de l'objet ayant été double-cliqué

Exemples

Le gestionnaire Double Clicked suivant est pour le table view (page 403) "messages" de l'application "Mail Search" distribuée avec AppleScript Studio. Le table view affiche les emails qui furent trouvés lors de la recherche. Double-cliquer un message sélectionné provoquera l'ouverture du message dans une fenêtre document.

```
on double clicked theObject
  set theController to controllerForWindow(window of theObject)
  if theController is not equal to null then
    tell theController to openMessages()
  end if
end double clicked
```

Ce gestionnaire utilise le paramètre `theObject` pour obtenir la fenêtre du table view ainsi il peut accéder au contrôleur de cette fenêtre. Le contrôleur est chargé d'exécuter les opérations de la fenêtre. Si le gestionnaire peut obtenir le contrôleur, il appelle le gestionnaire `openMessages` du contrôleur pour véritablement ouvrir le message sélectionné (ou les messages).

Version

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" s'appelait "Watson".

`end editing`

Appelé avant que l'édition ne se finisse.

Vous utiliserez généralement ce gestionnaire avec les objets text field (page 325), text view (page 569) ou autres. Le gestionnaire ne peut pas annuler la fin de l'édition, mais peut la préparer.

Pour plus d'informations, voir begin editing (page 346).

Syntaxe

`end editing` *reference* obligatoire

Paramètres

reference

La référence de l'objet text field (page 325), text view (page 569) ou autres pour lequel l'édition se finira

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire End Editing à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer la fin de l'édition.

```
on end editing theObject
  (* Perform operations here before editing stops. *)
end end editing
```

selection changed

Appelé après que la sélection d'un objet change.

Le gestionnaire peut exécuter des actions en réponse au changement de sélection. Généralement utilisé avec les "data views", comme browser (page 363), outline view (page 392) ou table view (page 403). Par exemple, lorsque l'utilisateur clique pour sélectionner une nouvelle rangée dans un objet outline view (page 392), le gestionnaire Selection Changed est appelé (s'il y en a un de connecter).

Syntaxe

`selection changed` *reference* obligatoire

Paramètres

reference

La référence de l'objet dont le gestionnaire Selection Changed est appelé

Exemples

Le gestionnaire Selection Changed suivant est extrait du fichier `Application.applescript` de l'application "Task List" distribuée avec AppleScript Studio (depuis la version 1.2). Ce gestionnaire est appelé chaque fois que la sélection dans un objet table view (page 403) change. S'il y a des rangées de sélectionner, ce gestionnaire appelle le gestionnaire `setUIValuesWithTaskValues` pour mettre à jour l'interface utilisateur fondée sur les rangées sélectionnées. Si aucune rangée n'est sélectionnée (la sélection a changé en aucune sélection), le gestionnaire appelle le gestionnaire `setDefaultUIValues` pour mettre à jour l'interface utilisateur à ses valeurs par défaut.

```
on selection changed theObject
  if name of theObject is "tasks" then
    -- If there is a selection then we'll update the UI;
    -- otherwise we set the UI to default values
    if (count of selected rows of theObject) > 0 then
      -- Get the selected data row of the table view
      set theTask to selected data row of theObject
      -- Update the UI using the selected task
      setUIValuesWithTaskValues(window of theObject, theTask)
    else
      -- Set the UI to default values
      setDefaultUIValues(window of theObject)
    end if
  end if
end selection changed
```

selection changing

Appelé très souvent pendant une opération de sélection multiple (comme la sélection de plusieurs rangées dans un objet table view (page 403) ou outline view (page 392)) où des éléments sont ajoutés ou supprimés de la sélection.

Le gestionnaire peut exécuter des actions en réponse au changement de sélection, bien qu'il ne devra pas exécuter de très longues opérations. Lorsque l'utilisateur conclut la sélection (en relâchant la souris par exemple), le gestionnaire selection changed (page 351) est appelé (s'il y en a un de

connecter).

Syntaxe

`selection changing`	*reference*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire Selection Changing est appelé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Selection Changing à un objet, AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre `theObject` référence l'objet, comme un objet browser (page 363), outline view (page 392) ou table view (page 403), pour lequel la sélection change. Vous pouvez utiliser ce gestionnaire pour exécuter des actions en réponse au changement de sélection.

```
on selection changing theObject
  -- Perform operations here in response to changing selection.
end selection changing
```

should begin editing

Appelé avant que l'édition ne commence.

Le gestionnaire peut retourner `false` pour annuler l'édition. Les classes comme text field (page 325) et text view (page 569) supportent ce gestionnaire.

Voir aussi begin editing (page 346) et should end editing (page 354).

Syntaxe

`should begin editing`	*reference*	obligatoire
`[object]`	*n'importe*	facultatif

Paramètres

reference

La référence de l'objet dont le gestionnaire Should Begin Editing est appelé

`[object]` *n'importe*

L'objet text field (page 325) ou text view (page 569) qui fera l'édition

Résultats

boolean

Retournera `true` pour autoriser le démarrage de l'édition ; `false` pour prévenir l'édition

Exemples

L'exemple suivant de gestionnaire Should Begin Editing appelle le gestionnaire `isItemEditable`, écrit par vous, pour déterminer s'il doit autoriser l'édition à commencer, puis retourne la valeur appropriée. Vous pourriez à la place exécuter des tests dans le gestionnaire lui-même ou vérifier une propriété.

```
on should begin editing theObject
  --Check property, perform test, or call handler to see if OK to edit
  set allowEditing to isItemEditable(theObject)
  return allowEditing
end should begin editing
```

should end editing

Appelé avant que l'édition ne se finisse.

Le gestionnaire peut retourner `false` pour annuler l'édition. Les classes comme text field (page 325) ou text view (page 569) supportent ce gestionnaire. Un usage commun est de ne pas autoriser la fin de l'édition si le texte courant est non valide.

Voir aussi end editing (page 350) et should begin editing (page 353).

Syntaxe

`should end editing`	*reference*	obligatoire
`[object]`	*n'importe*	facultatif

Paramètres

reference

La référence de l'objet dont le gestionnaire Should End Editing est appelé

`[object]` *n'importe*

L'objet text field (page 325) ou text view (page 569) qui fait l'édition

Résultats

boolean

Retournera `true` pour autoriser la fin de l'édition ; `false` pour continuer l'édition

Exemples

L'exemple suivant de gestionnaire Should End Editing appelle le gestionnaire `isItemValid`, écrit par vous, pour déterminer s'il doit autoriser la fin de l'édition (si l'élément d'édition est valide), puis retourne la valeur appropriée. Vous pourriez à la place exécuter une validation dans le gestionnaire lui-même ou vérifier une propriété.

```
on should end editing theObject
  --Check property, perform test, or call handler to see if OK
  --to stop editing (if edited item is valid)
  set allowStopEditing to isItemValid(theObject)
  return allowStopEditing
end should end editing
```

will dismiss

Appelé avant que l'objet soit rejeté.

Ce gestionnaire est supporté par les objets combo box (page 274). Le gestionnaire ne peut pas annuler le rejet, mais peut le préparer.

Syntaxe

`will dismiss` *reference* obligatoire

Paramètres

reference

La référence de l'objet combo box (page 274) dont le gestionnaire Will Dismiss est appelé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Dismiss à un objet combo box (page 274), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Vous pouvez utiliser ce gestionnaire pour préparer le rejet.

```
on will dismiss theObject
  (* Perform any operations to prepare for being dismissed. *)
end will dismiss
```

will pop up

Appelé avant qu'un menu déroulant ne surgisse.

Ce gestionnaire est supporté par les objets combo box (page 274) et popup button (page 301). Le gestionnaire ne peut pas annuler le surgissement, mais peut le préparer.

Syntaxe

`will pop up` *reference* obligatoire

Paramètres

reference

La référence de l'objet combo box (page 274) ou popup button (page 301) dont le gestionnaire Will Pop Up est appelé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Pop Up à un objet combo box (page 274), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre `theObject` référence l'objet combo box (page 274) ou popup button (page 301) qui est sur le point de surgir. Vous pouvez utiliser ce

gestionnaire pour préparer le surgissement (comme vérifier les éléments du combo box ou du popup button).

```
on will pop up theObject
  (* Perform any operations to prepare for popping up. *)
end will pop up
```

Cinquième partie

Data View Suite

Cette partie décrit la terminologie de la suite Data View d'AppleScript Studio.

La suite Data View définit les classes dont la principale caractéristique est d'afficher des rangées et des colonnes de données. La plupart des classes de la suite Data View hérite de la classe view (page 229) ou de la classe cell (page 264). La suite Data View définit aussi plusieurs Events pour le travail avec les éléments, cellules, rangées et colonnes trouvables dans les objets table view (page 403) et outline view (page 392).

Les classes, commandes et Events de la suite Data View sont décrits dans les chapitres suivants :

Classes 363
Commandes 417
Events 423

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Data View contient les classes suivantes :

browser 363
browser cell 370
data cell 372
data column 376
data item 379
data row 384
data source 386
outline view 392
table column 398
table header cell 401
table header view 402
table view 403

`browser`

Pluriel : `browsers`
Hérite de : `control` (page 280)
Classe Cocoa : NSBrowser

Fournit une interface utilisateur pour l'affichage et la sélection d'éléments depuis une liste de données, ou depuis des listes de données organisées hiérarchiquement comme un répertoire de chemins.

Lorsque vous travaillez avec une hiérarchie de données, les niveaux sont affichés dans des colonnes, lesquelles sont numérotées de gauche à droite. Les numéros des colonnes sont basés sur 0.

Vous trouverez l'objet browser dans le panneau "Cocoa-Data" d'Interface Builder. Vous pouvez régler la plupart de ses attributs dans la fenêtre Info d'Interface Builder.

L'illustration 5.1 montre un objet browser affichant les fichiers du disque dur. Pour plus d'informations, voir browser cell (page 370) et update (page 119), ainsi que Browsers dans la documentation Cocoa.

Note : Contrairement aux autres data views comme outline view (page 392) et table view (page 403), vous ne pourrez pas alimenter un objet browser avec un objet data source (page 386). En conséquence de quoi, les performances pourront être insuffisantes pour des objets browser affichant plus qu'un petit nombre d'éléments, aussi vous devrez préférer l'utilisation d'une des deux autres data views, bien sûr si cela est compatible avec vos impératifs.

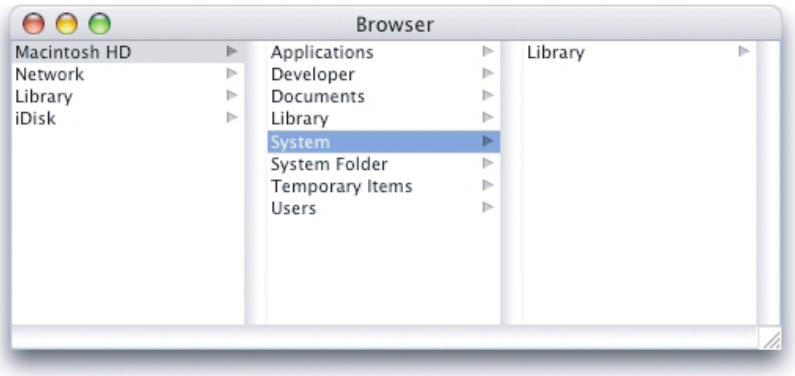

FIG. 5.1 - *Un browser view affichant une partie du système de fichiers*

Propriétés des objets de la classe Browser

En plus des propriétés qu'il hérite de la classe control (page 280), un objet browser possède ces propriétés :

accepts arrow keys

Accès : lecture / écriture
Classe : *boolean*
L'objet browser accepte-t-il la saisie avec les flèches de direction ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans

la fenêtre Info d'Interface Builder

allows branch selection

Accès : lecture / écriture
Classe : *boolean*
L'objet browser autorise-t-il la sélection d'une "branche" lorsque la sélection multiple est activée ? Chaque cellule peut être, soit une "branche" (comme un répertoire) ou une "feuille" (comme un fichier) ; par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

allows empty selection

Accès : lecture / écriture
Classe : *boolean*
Peut-il y avoir une sélection vide ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

allows multiple selection

Accès : lecture / écriture
Classe : *boolean*
Peut-il y avoir plusieurs éléments de sélectionner ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

cell prototype

Accès : lecture / écriture
Classe : *browser cell* (page 370)
La cellule prototype de l'objet browser ; cette cellule est copiée pour afficher les éléments dans les colonnes de l'objet browser

displayed cell

Accès : lecture / écriture
Classe : *browser cell* (page 370)
La cellule affichée dans l'objet browser

first visible column

Accès : lecture / écriture
Classe : *integer*
L'index de la première colonne visible de l'objet browser ; voir aussi la propriété *maximum visible columns*

has horizontal scroller

Accès : lecture / écriture
Classe : *boolean*
L'objet browser a-t-il un ascenseur horizontal ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

last column

Accès : lecture / écriture
Classe : *integer*
L'index de la dernière colonne de l'objet browser

last visible column

Accès : lecture / écriture
Classe : *integer*
L'index de la dernière colonne courante visible dans l'objet browser ; voir aussi la propriété *maximum visible columns*

loaded

Accès : lecture uniquement
Classe : *boolean*
Les données de l'objet browser sont-elles chargées ? Si `true`, toutes les informations des colonnes courantes affichées ont été acquises (un objet browser view appellera pour chaque cellule le gestionnaire will display browser cell (page 446) lorsqu'il aura besoin d'afficher les données dans une colonne)

maximum visible columns

Accès : lecture / écriture
Classe : *integer*
Le nombre total de colonnes visibles ; vous pouvez régler ce nombre dans la fenêtre Info d'Interface Builder

minimum column width

Accès : lecture / écriture
Classe : *real*
La largeur maximale de la colonne

path

Accès : lecture / écriture
Classe : *Unicode text*

Le chemin représentant l'élément sélectionné (par exemple, lorsque l'objet browser affiche des fichiers dans le système de fichiers)

path separator

Accès : lecture / écriture
Classe : *Unicode text*
La chaîne de caractères à utiliser comme séparateur pour les chemins ; par défaut, cette propriété est réglée sur le caractère slash (/)

reuses columns

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet browser réutilise ses colonnes ? Par défaut, cette propriété vaut `true`, signifiant que l'objet browser n'est pas libre de sauter à la colonne suivante lorsque les colonnes sont déchargées

selected cell

Accès : lecture / écriture
Classe : *browser cell* (page 370)
La cellule courante sélectionnée

selected column

Accès : lecture / écriture
Classe : *integer*
L'index de la colonne courante sélectionnée

send action on arrow key

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet browser exécute des actions lorsqu'il reçoit une entrée avec les flèches de direction ?

separates columns

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les colonnes soient séparées par des bordures ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder, mais uniquement si vous avez décoché la case "Is titled"

title height

Accès : lecture uniquement

Classe : *real*
La hauteur du titre

titled

Accès : lecture / écriture
Classe : *boolean*
L'objet browser utilise-t-il des titres ? Vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder ; non supportée dans la version 1.3 d'AppleScript Studio ; toutefois, l'instruction suivante pallie cette incompatibilité en utilisant la commande call method (page 94) :

```
set isTitled to call method "isTitled" of browser 1
```

uses title from previous column

Accès : lecture / écriture
Classe : *boolean*
L'objet browser devra-t-il utiliser la valeur de la colonne précédente comme titre de la prochaine colonne ? Par défaut, cette propriété vaut `true`

Éléments des objets de la classe Browser

En plus des éléments qu'il hérite de la classe control (page 280), un objet browser peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`cell` (page 264)
spécifier par : "Les formes-clés standards" (page 15)
Les cellules de l'objet browser ; peuvent généralement être de la classe browser cell (page 370)

Commandes supportées par les objets de la classe Browser

Votre script peut envoyer les commandes suivantes à un objet browser :

`path for` (page 112)

`update` (page 119)

Events supportés par les objets de la classe Browser

Un objet browser supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)

Browser view

`number of browser rows` (page 436)
`will display browser cell` (page 446)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

L'instruction suivante montre comment régler la propriété *path separator* d'un objet browser nommé "browser" dans une fenêtre nommée "main".

```
set path separator of browser "browser" of window "main" to ":"
```

Cette instruction est extraite du gestionnaire launched (page 135) de l'application "Browser" distribuée avec AppleScript Studio. Le gestionnaire complet est listé dans la section "Exemples" de la commande update (page 119).

Travailler avec les objets browser view est une tâche complexe ne pouvant pas être couverte en détails ici. L'application "Browser" fournit un exemple complet mais relativement simple permettant de naviguer dans le système de fichiers, d'afficher les fichiers et les dossiers dans une fenêtre identique à la vue en colonne de l'application Finder.

Version

Le support des Events de Glisser-Déposer est apparu dans la version 1.2 d'AppleScript Studio.

La propriété *titled* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

browser cell

Pluriel : `browser cells`
Hérite de : `cell` (page 264)
Classe Cocoa : NSBrowserCell

Représente la sous-classe par défaut d'une cellule utilisée pour afficher les données dans les colonnes d'un objet browser (page 363).

Chaque colonne d'un browser contient un objet matrix (page 289) rempli avec les cellules de l'objet browser. Pour plus d'informations, voir browser (page 363), ainsi que Browsers dans la documentation Cocoa.

Propriétés des objets de la classe Browser Cell

En plus des propriétés qu'il hérite de la classe cell (page 264), un objet browser cell possède ces propriétés :

alternate image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image alternée qui devra être utilisée par l'état "illuminé" de l'objet browser cell

leaf

Accès : lecture / écriture
Classe : *boolean*
Est-ce une cellule "feuille" ? Chaque cellule peut être, soit une "branche" (comme un répertoire) ou une "feuille" (comme un fichier) ; une cellule "branche" a une image sur son côté droit indiquant qu'elle contient quelquechose, des informations imbriquées sont disponibles ; une cellule "feuille" n'a pas d'image, indiquant que l'utilisateur a atteint une pièce d'information terminale

loaded

Accès : lecture / écriture
Classe : *boolean*
La cellule est-elle chargée ? `true` si l'état de la cellule a été réglé et qu'elle est prête pour l'affichage (un objet browser view appellera pour chaque cellule le gestionnaire will display browser cell (page 446) lorsqu'il aura besoin d'afficher les données dans une colonne)

Events supportés par les objets de la classe Browser Cell

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire will display browser cell (page 446) suivant est extrait de l'application "Browser" distribuée avec AppleScript Studio. L'application "Browser" sert à naviguer dans le système de fichiers, affichant les fichiers et les dossiers dans une fenêtre similaire à la fenêtre en mode colonne de l'application Finder. Ce gestionnaire utilise le Finder pour obtenir les infor-

mations sur les cellules de l'objet browser (page 363) devant être affichées, puis règle les propriétés de la cellule. Voir l'application "Browser" pour un exemple plus complexe de travail avec un objet browser view, y compris le code complet de ce gestionnaire.

```
on will display browser cell theObject row theRow browser cell theCell in
column theColumn
  -- Code to set the values of the cellContents and isLeaf variables
  -- is not shown
  set string value of theCell to cellContents
  set leaf of theCell to isLeaf
end will display browser cell
```

Notez que contrairement aux autres data views comme outline view (page 392) et table view (page 403), vous ne pourrez pas alimenter en données un objet browser avec un objet data source (page 386). En conséquence de quoi, les performances pourront être insuffisantes pour des objets browser affichant plus qu'un petit nombre d'éléments, aussi vous devrez préférer l'utilisation d'une des deux autres data views, bien sûr si cela est compatible avec vos impératifs.

data cell

Pluriel : `data cells`
Hérite de : `personne`
Classe Cocoa : ASKDataCell

Représente une cellule d'une ligne d'un objet data source (page 386).

Les objets data cell stockent le contenu des cellules, ainsi que d'autres informations pour accéder à leur données.

Vous créerez généralement un objet data source (page 386) pour gérer les données d'un objet outline view (page 392) ou table view (page 403). Vous créerez alors chaque objet data column (page 376) et leur fournirez un nom. Ce processus est montré dans la section "Exemples" de la commande append (page 417).

Puis vous créerez des objets data row (page 384) (pour un objet table view) ou des objets data item (page 379) (pour un objet outline view) pour la data source. Pour chaque data row ou data item créé, l'objet data source créera automatiquement des objets data cell pour chaque colonne, donnant

par défaut à chaque objet data cell le nom de sa colonne. Après création d'une ligne (ou d'un élément), vous pouvez régler les données de son objet data cell, généralement en spécifiant la ligne (ou l'élément) et le nom de la cellule. Vous pouvez utiliser ces mêmes informations pour obtenir le contenu d'un objet data cell.

Propriétés des objets de la classe Data Cell

Un objet data cell possède ces propriétés :

content

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu de la cellule ; synonyme de *contents*

contents

Accès : lecture / écriture
Classe : *item* (page 60)
Le contenu de la cellule ; synonyme de *content*

name

Accès : lecture / écriture
Classe : *Unicode text*
Le nom de la cellule ; lorsque vous créez un objet data row (page 384), un objet data cell est créé pour chaque colonne et, par défaut, le nom est réglé sur le nom de la colonne

Éléments des objets de la classe Data Cell

Un objet data cell peut uniquement contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`data item` (page 379)
spécifier par : "Les formes-clés standards" (page 15)
Les éléments des lignes que contient l'objet data cell

`data row` (page 384)
spécifier par : "Les formes-clés standards" (page 15)
Les lignes de l'objet data cell

Events supportés par les objets de la classe Data Cell

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire `getContactInfo` suivant est extrait de l'application "Table" distribuée avec AppleScript Studio. Vous le trouverez dans le fichier script `WithDataSource.applescript`. Ce gestionnaire montre comment régler le contenu d'un objet data cell avec le contenu d'un objet text field (page 325). Chaque data cell est identifiée par son nom, fournie plus tôt lorsque la data source fut initialisée.

```
-- Get the values from the text fields and set the cells in the data row
--
on getContactInfo(theWindow, theRow)
  tell theWindow
    set contents of data cell "name" of theRow
      to contents of text field "name"
    set contents of data cell "address" of theRow
      to contents of text field "address"
    set contents of data cell "city" of theRow
      to contents of text field "city"
    set contents of data cell "state" of theRow
      to contents of text field "state"
    set contents of data cell "zip" of theRow
      to contents of text field "zip"
  end tell
end getContactInfo
```

L'illustration 5.2 montre l'application "Table" lancée, avec un contact. Un autre gestionnaire de cette application est visible dans la section "Exemples" de la classe data column (page 376).

Pour obtenir des informations sur les data cell, étant connu la data row contenant la cellule, vous utiliserez une instruction comme celle ci-dessous, extraite du gestionnaire `setContactInfo` de la même application "Table". Comme dans l'exemple précédent, cette instruction apparaît à l'intérieur d'un bloc `tell` spécifiant la fenêtre :

```
set contents of text field "name"
```

```
  to contents of data cell "name" of theRow
```

L'application "Task List", disponible depuis la version 1.2 d'AppleScript Studio, contient un gestionnaire data representation (page 465) montrant comment accéder à tous les objets data cell d'un objet data source (page 386).

```
on data representation theObject of type ofType
  -- Set some local variables to various objects in the UI
  set theWindow to window 1 of theObject
  set theDataSource to data source of table view "tasks"
    of scroll view "tasks" of theWindow
  set theTasks to contents of every data cell of every data row
    of theDataSource
  set theSortColumn to sort column of theDataSource

  -- Statements for working with data cells not shown.
end data representation
```

Pour obtenir le contenu de chaque objet data cell d'une table, vous pouvez utiliser cette instruction :

```
set theData to contents of every data cell of every data row
  of data source of table view 1 of scroll view 1 of window "main"
```

Pour obtenir le contenu de chaque objet data cell d'une colonne particulière, vous pouvez utiliser cette instruction :

```
set theData to contents of data cell "zip" of data rows of dataSource
  of table  view "bigTable" of scroll view "myScrollView" of window 1
```

Note : Vous pouvez spécifier les objets par leur index, leur nom ou par un mélange des deux, comme dans les deux exemples précédents.

Pour un exemple utilisant les data cell avec des data item dans un objet outline view (page 392), voir la section "Exemples" de la classe data item (page 379). Pour un exemple qui extrait le nom à partir d'un objet data cell d'une ligne cliquée, voir la section "Exemples" de la classe table view (page 403).

Version

La propriété *content* est apparue avec la version 1.2 d'AppleScript Studio. Vous pouvez utiliser au choix *content* et *contents*, sauf à l'intérieur d'un gestionnaire d'Events, `contents of theObject` retournant une référence à l'objet plutôt que son contenu courant. Pour obtenir dans un gestionnaire d'Events le contenu d'un objet (comme le texte contenu dans un text field (page 325)), vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

Pour un exemple de script montrant la différence entre *content* et *contents*, voir la section "Version" de la classe control (page 280).

data column

Pluriel : `data columns`
Hérite de : `personne`
Classe Cocoa : ASKDataClumn

Représente une colonne dans un objet data source (page 386).

Cet objet stocke le nom de la colonne, la data source et d'autres informations sur la colonne. Vous pouvez utiliser les éléments d'un objet data column pour accéder à ses lignes ou aux cellules individuelles qui fournissent ses données.

Vous créerez généralement un objet data source (page 386) pour gérer les données d'un objet outline view (page 392) ou table view (page 403). Vous créerez alors chaque objet data column et leur fournirez un nom. Ce processus est montré dans la section "Exemples" de la commande append (page 417).

Pour un objet outline view (page 392), la colonne numéro un est la colonne "outline", laquelle contient les triangles de développement permettant l'expansion ou la contraction de ses éléments. La première colonne est généralement la colonne "outline" (bien que vous pouvez spécifier si vous autorisez l'utilisateur à réorganiser les colonnes dans la fenêtre Info d'Interface Builder).

Pour des informations de même nature, voir la classe data cell (page 372).

Propriétés des objets de la classe Data Column

Un objet data column possède ces propriétés (voir la section “Version” de cette classe pour savoir dans quelle version d’AppleScript Studio sont apparues certaines propriétés) :

data source

Accès : lecture uniquement
Classe : *data source* (page 386)
L’objet data source avec laquelle est associé l’objet data column

name

Accès : lecture / écriture
Classe : *Unicode text*
Le nom de la colonne

sort case sensitivity

Accès : lecture / écriture
Classe : *une des constantes* de sort case sensitivity (page 187)
La sensibilité du tri (sensible à la casse ou pas)

sort order

Accès : lecture / écriture
Classe : *une des constantes* de sort order (page 187)
L’ordre du tri (ascendant ou descendant)

sort type

Accès : lecture / écriture
Classe : *une des constantes* de sort type (page 188)
Le type de tri (alphabétique, numérique)

Éléments des objets de la classe Data Column

Un objet data column peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans “Les formes-clés standards” (page 15).

`data cell` (page 372)
spécifier par : “Les formes-clés standards” (page 15)
Les objets data cell de la colonne, un par ligne ; par défaut, il n’y a qu’un seul objet data cell pour chaque objet data column dans un objet data row

`data row` (page 384)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data row de la colonne

Events supportés par les objets de la classe Data Column

Cette classe n'est pas accessible dans AppleScript Studio, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire will open (page 165) suivant, extrait de l'application "Table" distribuée avec AppleScript Studio, montre comment créer et nommer les objets data column d'un objet data source (page 386). Ce gestionnaire fait ce qui suit :

- Obtient la référence de la data source à partir du paramètre `theObject` transmis au gestionnaire. La data source est une propriété de l'objet table view (page 403) résidant sur l'objet scroll view (page 213) de la fenêtre.
- Il indique à l'objet data source de créer 5 nouvelles colonnes, chacune avec le nom différent d'un champ de données d'un contact (nom, adresse, ville, etc...).

```
on will open theObject
  -- Set up reference variable to simplify later statements.
  set contactsDataSource to data source of table view "contacts"
    of scroll view "contacts" of theObject
  -- Add the data columns to the data source of the contacts table view.
  tell contactsDataSource
    make new data column at the end of the data columns
      with properties {name:"name"}
    make new data column at the end of the data columns
      with properties {name:"address"}
    make new data column at the end of the data columns
      with properties {name:"city"}
    make new data column at the end of the data columns
      with properties {name:"state"}
    make new data column at the end of the data columns
      with properties {name:"zip"}
  end tell
```

```
end will open
```

L'illustration 5.2 montre l'application en fonctionnement, avec un contact. Pour un autre exemple utilisant les objets data column, voir la section "Exemples" de la classe data item (page 379).

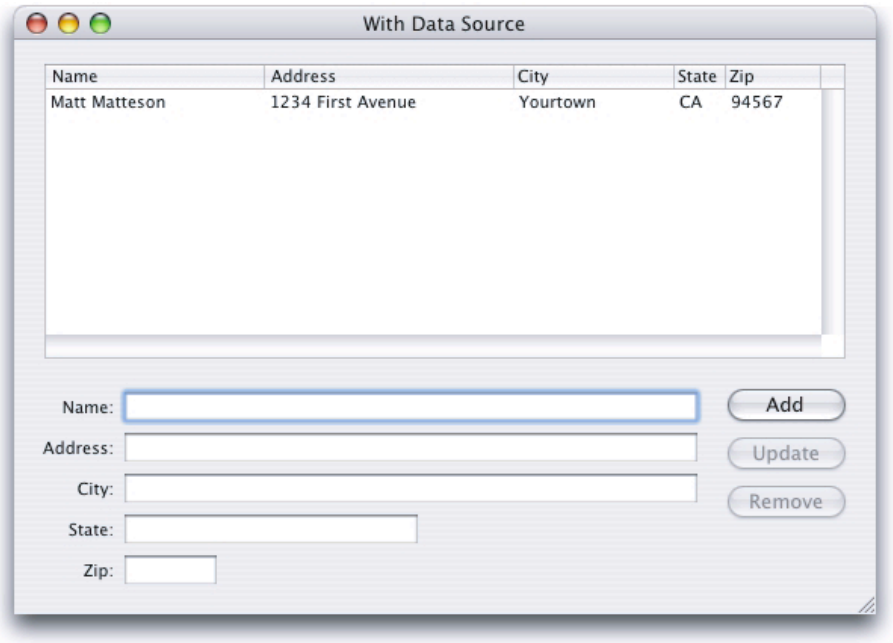

FIG. 5.2 - *L'application "Table"*

Version

Les propriétés *sort case sensitivity*, *sort order* et *sort type* furent ajoutées dans la version 1.2 d'AppleScript Studio.

data item

Pluriel : `data items`
Hérite de : `data row` (page 384)
Classe Cocoa : ASKDataItem

Représente une ligne d'une data source.

Un objet data item peut contenir des objets data item imbriqués, supporter le stockage des éléments dans une view hiérarchique, comme un objet

outline view (page 392), où l'utilisateur peut ouvrir un élément pour afficher les éléments contenus. Les propriétés d'un objet data item spécifient s'il a des objets data item imbriqués, ainsi que s'il a un élément parent, et si oui, la référence de l'élément parent. Ses éléments stockent n'importe quel objet data item imbriqué.

Vous créerez généralement un objet data source (page 386) pour gérer les données d'un objet outline view (page 392) ou table view (page 403). Vous créerez alors chaque objet data column (page 376) et leur fournirez un nom. Ce processus est montré dans la section "Exemples" de la commande append (page 417).

Puis, pour un objet outline view, vous créerez des objets data item pour la data source. Pour chaque data item créé, l'objet data source créera automatiquement un objet data cell (page 372) pour chaque colonne, donnant par défaut à chaque data cell le nom de sa colonne. Après la création d'un objet data item, vous pouvez régler les données de ses objets data cell, généralement en spécifiant le data item et le nom de la cellule. Vous pouvez utiliser ces mêmes informations pour obtenir le contenu d'un objet data cell.

Pour un objet table view (page 403), vous créerez des objets data row (page 384) à la place des objets data item, comme le décrit la section "Exemples" de la classe data row (page 384).

Propriétés des objets de la classe Data Item

En plus des propriétés qu'il hérite de la classe data row (page 384), un objet data item possède ces propriétés :

has data items

Accès : lecture uniquement
Classe : *boolean*
Cet élément contient-il des objets data item ?

has parent data item

Accès : lecture uniquement
Classe : *boolean*
Cet élément a-t-il un élément parent ?

parent data item

Accès : lecture / écriture
Classe : *data item* (page 379)
L'élément parent de l'élément

Éléments des objets de la classe Data Item

Un objet data item peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`data cell` (page 372)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data cell de la colonne, un par ligne

`data item` (page 379)
spécifier par : "Les formes-clés standards" (page 15)
Les éléments imbriqués des éléments

`data row` (page 384)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data row de la colonne

Events supportés par les objets de la classe Data Item

Cette classe n'est pas accessible dans AppleScript Studio, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire launched (page 135) suivant montre les étapes courantes pour travailler avec les objets data item et data column dans une data source, comprenant :

- création de l'objet data source (page 386) et stockage de cet objet comme un élément de l'objet `application`
- création et baptême des objets data column (page 376) pour la data source
- création des objets data item parents et réglage du contenu de leurs objets data cell (page 372)
- création de nouveaux objets data item en tant que fils de l'objet data item parent et réglage de leurs contenus
- création d'objets data item fils supplémentaires
- assignation de la data source à une propriété de l'objet outline view (page 392)

Le gestionnaire launched est appelé à la fin de la séquence de démarrage d'une application, aussi c'est une bonne place pour créer une data source pour un objet outline view et la peupler avec des objets data item. Ce gestionnaire ajoute les informations suivantes à l'objet outline view :

```
- Things to do
  - Work on outline example
    - Make it plain and simple
    - Put it all in a "launched" event handler
  - Put it in my iDisk when done
```

Voici le code du gestionnaire launched :

```
on launched theObject
  -- Create the data source; this places it in the application
  -- object's data source elements. (Assign it to outline view below.)
  set dataSource to make new data source at end of data sources
    with properties {name:"tasks"}

  -- Create the data columns
  tell dataSource
    make new data column at end of data columns
      with properties {name:"task"}
    make new data column at end of data columns
      with properties {name:"completed"}
  end tell

  -- Create the top-level parent data item "Things to do"
  set parentItem to make new data item at end of data items of dataSource
  set contents of data cell "task" of parentItem to "Things to do"
  set contents of data cell "completed" of parentItem to "--"

  -- Create the first child data item "Work on outline example", which
  -- will have its own children
  set childItem to make new data item at end of data items of parentItem
  set contents of data cell "task" of childItem
    to "Work on outline example"
  set contents of data cell "completed" of childItem to "Yes"

  -- Create first child data item of "Work on outline example"
```

```
  set childChildItem to make new data item at end of data items of childItem
  set contents of data cell "task" of childChildItem
    to "Make it plain and simple"
  set contents of data cell "completed" of childChildItem to "Yes"

  -- Create second child data item of "Work on outline example"
  set childChildItem to make new data item at end of data items
    of childItem
  set contents of data cell "task" of childChildItem
    to "Put it all in a \"launched\" event handler"
  set contents of data cell "completed" of childChildItem to "Yes"

  -- Create the second child data item of "Things to do"
  set childItem to make new data item at end of data items of parentItem
  set contents of data cell "task" of childItem
    to "Put it in my iDisk when done"
  set contents of data cell "completed" of childItem to "No"

  -- Assign the data source to the outline view
  set data source of outline view "tasks" of scroll view "scroll"
    of window "main" to dataSource
end launched
```

L'illustration 5.3 montre l'application lancée, avec tous les objets data item développés.

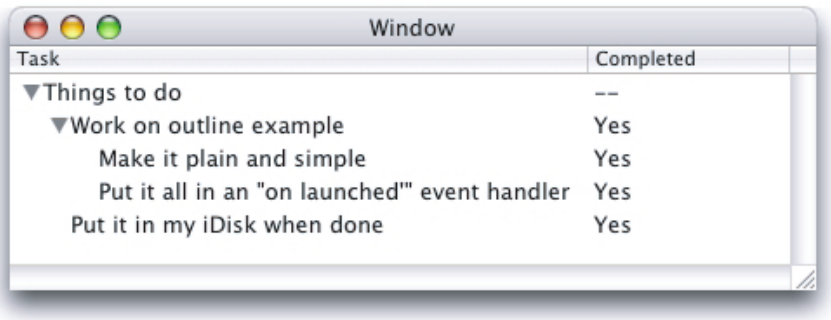

FIG. 5.3 - *L'application "To Do list"*

data row

Pluriel : `data rows`
Hérite de : `personne`
Classe Cocoa : ASKDataRow

Représente une ligne dans un objet data source (page 386).

Cet objet stocke la ligne d'une data source et d'autres informations. Vous pouvez utiliser les éléments d'un objet data row pour accéder à ses colonnes ou aux cellules individuelles qui fournissent ses données.

Vous créerez généralement un objet data source (page 386) pour gérer les données d'un objet outline view (page 392) ou table view (page 403). Vous créerez alors chaque objet data column (page 376) et leur fournirez un nom. Ce processus est montré dans la section "Exemples" de la commande append (page 417).

Puis, pour un objet table view, vous créerez des objets data row pour la data source.

Important
Vous devez d'abord créer les objets data column avant de créer les objets data row.

Pour chaque ligne créée, l'objet data source créera automatiquement un objet data cell (page 372) pour chaque colonne, donnant par défaut à chaque objet data cell le nom de sa colonne. Après création d'une ligne, vous pouvez régler les données de ses objets data cell, généralement en spécifiant la ligne et le nom de la cellule. Vous pouvez utiliser ces mêmes informations pour obtenir le contenu d'un objet data cell.

Pour un objet outline view (page 392), vous créerez des objets data item (page 379) à la place des objets data row, comme le décrit la section "Exemples" de la classe data item (page 379).

Propriétés des objets de la classe Data Row

Un objet data row possède ces propriétés :

associated object

Accès : lecture / écriture
Classe : *item* (page 60)
Un objet pouvant être associé avec l'objet data row

data source

Accès : lecture uniquement
Classe : *data source* (page 386)
L'objet data source avec lequel est associé l'objet data row

Éléments des objets de la classe Data Row

Un objet data row peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`data cell` (page 372)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data cell de l'objet data row ; chaque cellule stocke son nom, son contenu et d'autres informations

`data column` (page 376)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data column de l'objet data row ; chaque colonne stocke son nom, la data source et d'autres informations

Events supportés par les objets de la classe Data Row

Cette classe n'est pas accessible dans AppleScript Studio, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

L'instruction suivante montre comment créer un objet data row. Cette instruction est extraite du gestionnaire clicked (page 348) de l'application "Table" distribuée avec AppleScript Studio.

```
set theRow to make new data row at the end of the data rows
  of contactsDataSource
```

Pour un exemple qui extrait le nom à partir d'un objet data cell dans une ligne cliquée, voir la section "Exemples" de la classe table view (page 403). Pour un exemple supplémentaire sur le travail avec les objets data row, voir la section "Exemples" de la classe data cell (page 372).

data source

Pluriel : `data sources`
Hérite de : `personne`
Classe Cocoa : ASKDataSource

Stocke les données et les fournit aux views affichant les lignes et les colonnes de données.

Un objet data source représente une forme de sauvegarde pour un tableau et est basé sur une classe spéciale fournie par le framework AppleScriptKit d'AppleScript Studio.

Pour des informations de même nature, voir les classes data cell (page 372), data column (page 376), data item (page 379) et data row (page 384), ainsi que Table Views dans la documentation Cocoa.

Votre application fournira un objet data source avec des lignes et des colonnes de données à une view comme table view (page 403) ou outline view (page 392). Une fois que vous avez fourni les données, l'objet data source travaille avec la view pour automatiquement afficher les bonnes informations en fonction des actions de l'utilisateur, par exemple, lorsque l'utilisateur scrolle, redimensionne la fenêtre, réorganise les colonnes ou modifie les lignes et les colonnes affichées.

Utiliser un objet data source est plus efficace que de fournir les données dans les gestionnaires devant être appelés pour chaque morceau de données. Et pour faire un usage encore plus efficace de l'objet data source, vous pouvez régler sa propriété *update views* sur `false` avant sa mise à jour, puis la régler sur `true` après pour que la mise à jour de la view associée se fasse d'un seul coup.

Chaque view affichant des lignes et des colonnes de données utilisera au maximum une seule data source. Toutefois, vous pouvez utiliser plusieurs views avec la même data source si, par exemple, vous souhaitez insister sur différents aspects des données. Puis si vous modifiez les données de la data source, chaque view sera mise automatiquement à jour afin de refléter les nouvelles valeurs.

Pour des exemples montrant comment créer un objet data source dans les scripts de votre application, voir les sections "Exemples" de la commande append (page 417) et de la classe data item (page 379). Vous pouvez aussi créer dans Interface Builder un objet data source en le glissant depuis le panneau "Cocoa-AppleScript". Ce mécanisme n'est pas recommandé et n'est pas décrit ici, mais vous pouvez le voir dans le tutoriel de l'application "Mail

Search" "*Building Applications With AppleScript Studio*".

Depuis la version 1.2 d'AppleScript Studio, les objets data source peuvent être triés. L'objet data source retiendra encore l'ordre dans lequel les lignes furent ajoutées à la data source, mais il pourra les présenter dans un ordre trié. Pour activer le tri d'un objet data source, vous devez faire ce qui suit :

- Ajoutez les propriétés suivantes lors de la création des objets data column dans l'instruction `make new`.
  - *sort order* : `ascending` ou `descending`
  - *sort type* : `alphabetical` ou `numerical`
  - *sort case sensitivity* : `case sensitive` ou `case insensitive`

  Par exemple :
  ```
  make new data column at end of data columns of theDataSource with
  properties {name: "name", sort order: ascending, sort type: alphabetical,
  sort case sensitivity: case sensitive}
  ```
- Réglez la propriété *sorted* de l'objet data source sur `true`.
  Par exemple :
  ```
  set sorted of theDataSource to true
  ```
- Réglez la propriété *sort column* de la data source sur la colonne initiale à trier.
  Par exemple :
  ```
  set sort column of theDataSource to data column "name" of theDataSource
  ```
- Connectez un gestionnaire column clicked (page 430) à votre objet table view (page 403). Cela fournira l'opportunité, lorsqu'un utilisateur cliquera dans l'en-tête (l'objet table header view (page 402)) de la colonne, de modifier la colonne sélectionnée pour le tri, ainsi que l'ordre de tri des données de la colonne dans votre data source. L'exemple de script montré dans la section "Exemples" de cette classe est relativement standard et peut être utilisé, tel quel, dans votre application.
  Pour des exemples complets, voir les applications "Table Sort" et "Task List" distribuées depuis la version 1.2 d'AppleScript Studio.

Si vous réglez le type de tri d'une colonne sur `numerical`, vous aurez besoin de vous assurer que le contenu des objets data cell de cette colonne sont bien en fait des nombres et non des chaînes de caractères. Ou si vous le réglez sur `alphabetical`, le contenu des objets data cell devra être des chaînes de caractères (les chiffres devront être au format `string` (`"23"`)).

Propriétés des objets de la classe Data Source

Un objet data source possède ces propriétés (voir la section “Version” de cette classe pour savoir dans quelle version d’AppleScript Studio sont apparues certaines propriétés) :

localized sort

Accès : lecture / écriture
Classe : *boolean*
Les données doivent-elles être triées en utilisant des règles localisées ?

sort column

Accès : lecture / écriture
Classe : *data column* (page 376)
l’objet data column dans lequel se fait le tri des données ; les propriétés de la colonne contrôle le tri

sorted

Accès : lecture / écriture
Classe : *boolean*
La data source doit-elle être triée ?

update views

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les views de la data source soient mises à jour ? Par efficacité, vous devrez éviter les mises à jour non nécessaires en réglant cette propriété sur `false` avant la mise à jour de la data source, puis la régler sur `true` après, comme il est montré dans la section “Exemples” de cette classe ; voir aussi la commande update (page 119)

Éléments des objets de la classe Data Source

Un objet data source peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans “Les formes-clés standards” (page 15).

`data cell` (page 372)
spécifier par : “Les formes-clés standards” (page 15)
Les objets data cell de l’objet data source ; chaque cellule stocke son nom, son contenu et d’autres informations

`data column` (page 376)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data column de l'objet data source ; chaque colonne stocke son nom et d'autres informations

`data item` (page 379)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data item de l'objet data source

`data row` (page 384)
spécifier par : "Les formes-clés standards" (page 15)
Les objets data row de l'objet data source

`view` (page 229)
spécifier par : "Les formes-clés standards" (page 15)
Les views de l'objet data source

Commandes supportées par les objets de la classe Data Source

Votre script peut envoyer la commande suivante à un objet data source :

`append` (page 417)

Events supportés par les objets de la classe Data Source

Un objet data source supporte les gestionnaires répondant aux Events suivants :

Nib

`awake from nib` (page 123)
Vous pouvez connecter un gestionnaire awake from nib à un objet data source uniquement si vous créez la data source dans Interface Builder, pas si vous la créez dans le fichier script (l'approche recommandée). Toutefois, il est peu probable que vous ayez besoin de connecter ce gestionnaire à un objet data source. Voir la description de cette classe plus haut pour plus d'informations sur la création d'un objet data source dans Interface Builder.

Exemples

Pour des exemples montrant comment créer un objet data source, voir les sections "Exemples" de la commande append (page 417) et de la classe data item (page 379).

Lorsque vous modifiez les données d'une data source dans une view visible, les performances risquent probablement de souffrir de la mise à jour continue de la view par la data source. Vous pouvez assurer des performances optimales en désactivant la mise à jour le temps que vous modifiez la data source, puis vous la réactivez de nouveau lorsque vous avez fini. Les lignes suivantes montrent comment faire cela :

```
-- Turn off updating
set update views of theDataSource to false
-- Add statements here that modify the data source
--
-- Turn updating back on
set update views of theDataSource to true
```

Le tri des data source, ajouté dans la version 1.2 d'AppleScript Studio, est décrit plus haut dans la description de cette classe. Vous pouvez aussi voir le tri dans les applications "Table" et "Task List" (disponibles depuis la version 1.2 d'AppleScript Studio).

Le gestionnaire column clicked (page 430) suivant montre comment gérer une modification faite dans la colonne sélectionnée pour le tri. Vous connecterez le gestionnaire Column Clicked à l'objet table view (page 403) ou outline view (page 392) contenant les colonnes à trier. Ce gestionnaire est relativement standard et peut être utilisé, tel quel, dans la plupart des applications. Il fait ce qui suit :

- Il obtient l'identificateur de la colonne cliquée.
- Il obtient la colonne courante servant pour le tri de la data source.
- Si les colonnes sont différentes, l'utilisateur a choisi une nouvelle colonne pour le tri, aussi il règle la propriété *sort column* de la data source sur la nouvelle colonne.
- Si les colonnes sont identiques, l'utilisateur a choisi un nouvel ordre de tri, aussi il modifie son ordre de tri (de `ascending` vers `descending`, ou vice versa).
- Il appelle la commande update (page 119) pour redessiner les données triées.

```
on column clicked theObject table column tableColumn
  -- Get the data source of the table view
  set theDataSource to data source of theObject
```

```
  -- Get the identifier of the clicked table column;
  -- you can instead use the name of the column
  set theColumnIdentifier to identifier of tableColumn

  -- Get the current sort column of the data source
  set theSortColumn to sort column of theDataSource

  -- If the current sort column is not the same as the clicked column
  --  then switch the sort column
  if (name of theSortColumn) is not equal to theColumnIdentifier then
    set the sort column of theDataSource to
      data column theColumnIdentifier of theDataSource
  else
    -- Otherwise change the sort order
    if sort order of theSortColumn is ascending then
      set sort order of theSortColumn to descending
    else
      set sort order of theSortColumn to ascending
    end if
  end if

  -- Update the table view (so it will be redrawn)
  update theObject
end column clicked
```

Les applications “Table Sort” et “Task List” fournissent des exemples complets sur la manière de trier une data source.

Travailler avec les objets data source est une tâche complexe ne pouvant pas être couverte en détails ici. Pour un exemple supplémentaire, relativement simple, voir l’application “Outline”, laquelle montre comment utiliser un objet outline view (page 392) pour afficher les éléments du système de fichiers. L’application “Table” utilise une data source avec un objet table view (page 403).

Pour un exemple plus détaillé, voir les chapitres du guide “*Inside Mac OS X : Building Applications With AppleScript Studio*” qui décrivent comment construire l’application “Mail Search” (laquelle est aussi distribuée avec AppleScript Studio).

Version

Le support des Events de glisser-déposer est apparu dans la version 1.2 d'AppleScript Studio.

Le tri des objets data source est apparu avec la version 1.2 d'AppleScript Studio.

Les propriétés *localized sort*, *sort column* et *sorted* furent ajoutées dans la version 1.2 d'AppleScript Studio.

Les applications "Table Sort" et "Task List" furent ajoutées dans la version 1.2 d'AppleScript Studio.

La méthode conseillée (et plus convenable) pour utiliser une data source est de la créer et de l'assigner directement dans le script de l'application, comme le montre la section "Exemples" de la commande append (page 417) ou de la classe data item (page 379). Cette option fut ajoutée dans la version 1.1 d'AppleScript Studio. Avant cette version, vous deviez utiliser un processus plus encombrant d'ajout et de connexion de la data source avec votre application dans Interface Builder.

Notez que si vous souhaitez connecter un gestionnaire awake from nib (page 123) à une data source, vous devrez ajouter et connecter la data source dans Interface Builder. Les étapes pour le faire sont décrites dans "*Inside Mac OS X : Building Applications With AppleScript Studio*", disponible dans l'aide de Project Builder.

L'élément `view` fut ajouté dans la version 1.1 d'AppleScript Studio.

`outline view`

Pluriel : `outline views`
Hérite de : `table view` (page 403)
Classe Cocoa : NSOutlineView

Une view qui utilise un format de lignes et de colonnes pour afficher des données hiérarchisées pouvant être développées ou contractées, comme des répertoires et des fichiers dans un système de fichiers.

L'utilisateur pourra développer et contracter les lignes, éditer les valeurs et redimensionner et réorganiser les colonnes.

L'illustration 5.4 montre un objet outline view affichant une hiérarchie de fichiers et de dossiers.

Vous trouverez l'objet outline view dans le panneau "Cocoa-Data" d'In-

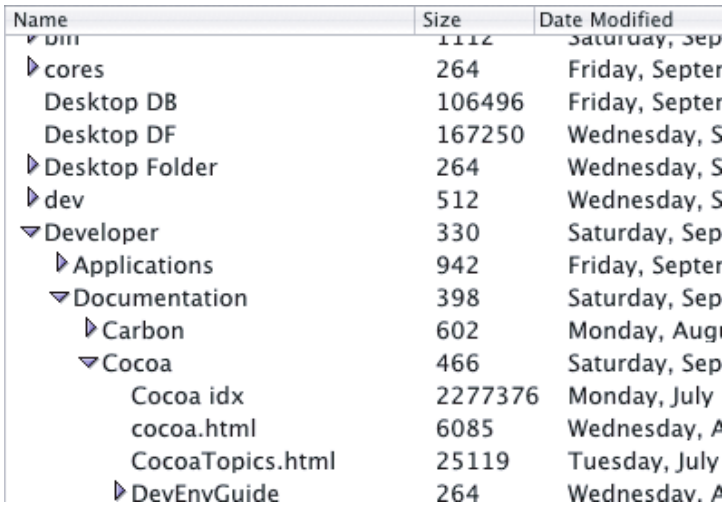

FIG. 5.4 - *Un objet outline view*

terface Builder. Vous pouvez régler la plupart de ses attributs dans la fenêtre Info d'Interface Builder.

Bien qu'AppleScript Studio fournissent des gestionnaires d'Events pour gérer les données qu'un objet outline view affiche, l'approche conseillée et largement plus efficace est d'utiliser un objet data source (page 386).

Pour plus d'informations sur les objets outline view, voir Outline Views dans la documentation Cocoa.

Propriétés des objets de la classe Outline View

En plus des propriétés qu'il hérite de la classe table view (page 403), un objet outline view possède ces propriétés :

auto resizes outline column

Accès : lecture / écriture
Classe : *boolean*
La colonne "outline" doit-elle être automatiquement redimensionnée ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

auto save expanded items

Accès : lecture / écriture
Classe : *boolean*
Non supportée dans la version 1.3 d'AppleScript Studio ; faut-il que

l'état développé des éléments "outline" soit automatiquement enregistré ? Par défaut, cette propriété vaut `false`

clicked data item

Accès : lecture uniquement
Classe : *data item* (page 379)
L'objet data item qui a été cliqué ; disponible depuis la version 1.3 d'AppleScript Studio ; retourne "missing value" si aucun élément n'a été cliqué, aussi vous devrez accéder à cette valeur dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

edited data item

Accès : lecture uniquement
Classe : *data item* (page 379)
L'objet data item qui a été édité ; disponible depuis la version 1.3 d'AppleScript Studio ; retourne "missing value" si aucun élément n'a été édité, aussi vous devrez accéder à cette valeur dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

indentation per level

Accès : lecture / écriture
Classe : *real*
La quantité d'indentation par niveau ; par défaut, cette propriété vaut 16.0

marker follows cell

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le marqueur suive les cellules (c'est à dire, comme les cellules de l'objet outline view sont indentés, faut-il que le triangle soit aussi indenté) ? Par défaut, cette propriété vaut `true`

selected data item

Accès : lecture / écriture
Classe : *data item* (page 379)
L'objet data item sélectionné ; disponible depuis la version 1.3 d'AppleScript Studio ; retourne "missing value" si aucun élément n'a été sélectionné, aussi vous devrez accéder à cette valeur dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section

"Exemples" de la commande path for (page 112))

selected data items

Accès : lecture / écriture
Classe : *list*
Les objets data item sélectionnés ; disponible depuis la version 1.3 d'AppleScript Studio ; retourne "missing value" si aucun élément n'a été sélectionné, aussi vous devrez accéder à cette valeur dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

outline table column

Accès : lecture / écriture
Classe : *table column* (page 398)
L'objet table column contenant le "outline"

Éléments des objets de la classe Outline View

Un objet outline view peut uniquement contenir les éléments qu'il hérite de table view (page 403).

Commandes supportées par les objets de la classe Outline View

Votre script peut envoyer les commandes suivantes à un objet outline view :

`item for` (page 420)
`update` (page 119)

Events supportés par les objets de la classe Outline View

Un objet outline view supporte les gestionnaires répondant aux Events suivants :

Action

`clicked` (page 348)
`double clicked` (page 349)

Data View

`column clicked` (page 430)
`column moved` (page 430)

column resized (page 431)
selection changed (page 351)
selection changing (page 352)
should select column (page 442)
should select item (page 443)
should select row (page 444)
should selection change (page 445)
will display cell (page 447)

Glisser-Déposer

conclude drop (page 481)
drag (page 483)
drag entered (page 483)
drag exited (page 484)
drag updated (page 485)
drop (page 486)
prepare drop (page 488)

Clavier

keyboard down (page 133)
keyboard up (page 134)

Souris

mouse down (page 137)
mouse dragged (page 138)
mouse entered (page 139)
mouse exited (page 139)
mouse up (page 141)
right mouse down (page 147)
right mouse dragged (page 149)
right mouse up (page 150)
scroll wheel (page 151)

Nib

awake from nib (page 123)

Outline View

change item value (page 427)

`child of item` (page 428)
`item expandable` (page 432)
`item value` (page 434)
`number of items` (page 437)
`should collapse item` (page 440)
`should expand item` (page 441)
`will display item cell` (page 449)
`will display outline cell` (page 450)

Table View

`cell value` (page 424)
`change cell value` (page 426)
`number of rows` (page 439)

View

`bounds changed` (page 243)

Exemples

Le script suivant montre comment identifier un objet outline view et lui envoyer une commande update (page 119).

```
tell outline view "outline" of scroll view "scroll" of window "main" to update
```

Travailler avec les objets outline view est une tâche complexe ne pouvant pas être couverte en détails ici. Pour un exemple complet mais relativement simple, voir la section "Exemples" de la classe data item (page 379). Pour un autre exemple plus détaillé, voir les chapitres du guide "*Building Applications With AppleScript Studio*" qui décrivent comment construire l'application "Mail Search" (laquelle est aussi distribuée avec AppleScript Studio).

Version

Les propriétés suivantes furent ajoutées dans la version 1.3 d'AppleScript Studio :

- *clicked data item*
- *edited data item*

- *selected data item*
- *selected data items*

Le support des Events de Glisser-Déposer fut ajouté dans la version 1.2 d'AppleScript Studio.

La propriété *auto save expanded items* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Voir la section "Version" de la classe table view (page 403) (de laquelle hérite la classe Outline View) pour les propriétés ajoutées dans la version 1.2 d'AppleScript Studio pour le support du tri.

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" s'appelait "Watson".

table column

Pluriel : `table columns`
Hérite de : `personne`
Classe Cocoa : NSTableColumn

Stocke les caractéristiques d'affichage et l'identificateur d'une colonne d'un objet table view (page 403) ou outline view (page 392).

L'objet table column détermine les limites en hauteur et en largeur, la possibilité de redimensionner et d'éditer ses colonnes dans les objets table view ou outline view. Il stocke aussi deux objets cell : le header cell, lequel est utilisé pour dessiner l'en-tête de la colonne, et son objet data cell (page 372) utilisé pour dessiner les valeurs de chaque ligne.

Lorsque vous insérez un objet table view ou outline view dans Interface Builder, la view contient automatiquement un objet table column pour chaque colonne. Si les objets table view ou outline view utilisent un objet data source (page 386), vous devrez spécifier, soit un nom identificateur, soit un nom AppleScript pour chaque colonne. Pour plus d'informations, voir la propriété *identifier* et la section "Version" de cette classe.

Important
Assurez vous bien que les noms AppleScript que vous avez assignés aux objets table column dans Interface Builder sont identiques aux propriétés *name* des objets data column de la data source de l'objet table view.

Pour plus d'informations, voir les classes table header cell (page 401) et table header view (page 402), ainsi que Table Views dans la documentation Cocoa.

Propriétés des objets de la classe Table Column

Un objet table column possède ces propriétés :

data cell

Accès : lecture / écriture
Classe : *data cell* (page 372)
Non supportée dans la version 1.3 d'AppleScript Studio ; l'objet data cell de la colonne

editable

Accès : lecture / écriture
Classe : *boolean*
La colonne est-elle éditable ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

header cell

Accès : lecture / écriture
Classe : *table header cell* (page 401)
L'objet table header cell utilisé pour dessiner l'en-tête de la colonne

identifier

Accès : lecture / écriture
Classe : *Unicode text*
Le nom utilisé par l'objet data source pour identifier une colonne ; vous pouvez régler cette valeur dans le champ "identifier" du panneau "Attributes" de la fenêtre Info d'Interface Builder ; voir la section "Version" de cette classe pour plus d'informations

maximum width

Accès : lecture / écriture
Classe : *real*
La largeur maximale de la colonne ; par défaut, cette propriété vaut 1000 ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

minimum width

Accès : lecture / écriture
Classe : *real*
La largeur minimale de la colonne ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder

resizable

Accès : lecture / écriture
Classe : *boolean*
La colonne est-elle redimensionnable ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

table view

Accès : lecture / écriture
Classe : *table view* (page 403)
L'objet table view ou outline view (cet objet hérite de table view) contenant l'objet table column

width

Accès : lecture / écriture
Classe : *real*
La largeur de la colonne

Éléments des objets de la classe Table Column

Un objet table column n'a aucun élément.

Events supportés par les objets de la classe Table Column

Un objet table column supporte les gestionnaires répondant aux Events suivants :

Nib

`awake from nib` (page 123)

Exemples

Pour un exemple montrant comment accéder aux propriétés d'un objet table column, voir le gestionnaire column clicked (page 430) de la section "Exemples" de la classe data source (page 386). La section "Exemples" de la classe table view (page 403) se concentre sur l'utilisation des objets table view, avec ou sans objet data source.

Version

Depuis la version 1.2 d'AppleScript Studio, et la version d'Interface Builder livrée avec Mac OS X version 10.2, vous pouvez nommer les objets table

column d'un objet table view (page 403) ou outline view (page 392) en utilisant le champ "Name" du panneau "AppleScript" de la fenêtre Info d'Interface Builder. Il n'est plus nécessaire de saisir la valeur dans le champ "Identifier" du panneau "Attributes" (bien que cela soit toujours supporté pour la compatibilité avec les versions plus anciennes).

Si vous spécifiez un nom identificateur (plutôt qu'un nom AppleScript) à un objet table column, il devra correspondre au nom de l'objet data column (page 376) de votre data source. Autrement, depuis la version 1.1 d'AppleScript Studio, aucune donnée ne sera fournie pour que l'objet data column soit dessiné.

La propriété *data cell* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

`table header cell`

Pluriel : `table header cells`
Hérite de : `text field cell` (page 330)
Classe Cocoa : NSTableHeaderCell

Utilisé pour un objet table header view pour dessiner les en-têtes de ses colonnes.

Pour plus d'informations, voir table header view (page 402), ainsi que Table Views dans la documentation Cocoa.

Propriétés des objets de la classe Table Header Cell

Un objet table header cell possède uniquement les propriétés qu'il hérite de la classe text field cell (page 330).

Events supportés par les objets de la classe Table Header Cell

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Vous ne scripterez généralement pas un objet table header cell, lequel n'ajoute rien à sa super-classe text field cell (page 330), ni propriétés, ni éléments.

table header view

Pluriel : table header views
Hérite de : view (page 229)
Classe Cocoa : NSTableHeaderView

Utilisé par un objet table view pour dessiner les en-têtes au-dessus de ses colonnes et pour gérer les Events Souris de ces en-têtes.

Pour plus d'informations, voir table view (page 403), ainsi que Table Views dans la documentation Cocoa.

Propriétés des objets de la classe Table Header View

En plus des propriétés qu'il hérite de la classe view (page 229), un objet table header view possède ces propriétés :

dragged column

Accès : lecture uniquement
Classe : *integer*
Si l'utilisateur fait glisser une colonne, le numéro d'index de cette colonne ; autrement -1

dragged distance

Accès : lecture uniquement
Classe : *real*
Si l'utilisateur fait glisser une colonne, la distance horizontale parcourue par la colonne depuis sa position initiale, autrement la valeur est sans importance

resized column

Accès : lecture uniquement
Classe : *integer*
Si l'utilisateur redimensionne une colonne, le numéro d'index de cette colonne ; autrement -1

table view

Accès : lecture / écriture
Classe : *table view* (page 403)
L'objet table view contenant l'objet table header view

Éléments supportés par les objets de la classe Table Header View

Un objet table header view peut uniquement contenir les éléments qu'il hérite de la classe view (page 229).

Events supportés par les objets de la classe Table Header View

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Vous ne scripterez généralement pas un objet table header view.

table view

Pluriel : `table views`
Hérite de : `control` (page 280)
Classe Cocoa : NSTableView

Une view qui affiche des enregistrements de données dans un tableau, et qui autorise l'utilisateur à éditer les valeurs, à redimensionner et à réorganiser les colonnes.

Un objet table view affiche les données d'une série d'enregistrements apparentés, avec les lignes représentant des enregistrements individuels et les colonnes les attributs de ces enregistrements. Si vous utilisez un objet data source (page 386) pour alimenter en données le table view, la data source fonctionnera avec la view pour automatiquement afficher les bonnes informations lorsque l'utilisateur scrollera, redimensionnera la fenêtre, réorganisera les colonnes, ou modifiera les lignes et les colonnes affichées.

Vous trouverez l'objet table view dans le panneau "Cocoa-Data" d'Interface Builder. Lorsque vous insérez un objet table view dans une fenêtre, cet objet est automatiquement inclus dans un objet scroll view (page 213). Vous pouvez régler la plupart des attributs des objets table view dans la fenêtre Info d'Interface Builder, mais pour faire cela, vous devrez double-cliquer pour sélectionner l'objet table view, et non sélectionner l'objet scroll view le contenant (le titre de la fenêtre Info d'Interface Builder devra mentionner "NSTableView").

L'illustration 5.5 montre un objet table view dans Interface Builder. Les informations des lignes et des colonnes sont temporairement remplies par

Interface Builder, lors de la compilation ces informations disparaîtront et seront remplacées par les vôtres. Pour plus d'informations, voir Table Views dans la documentation Cocoa.

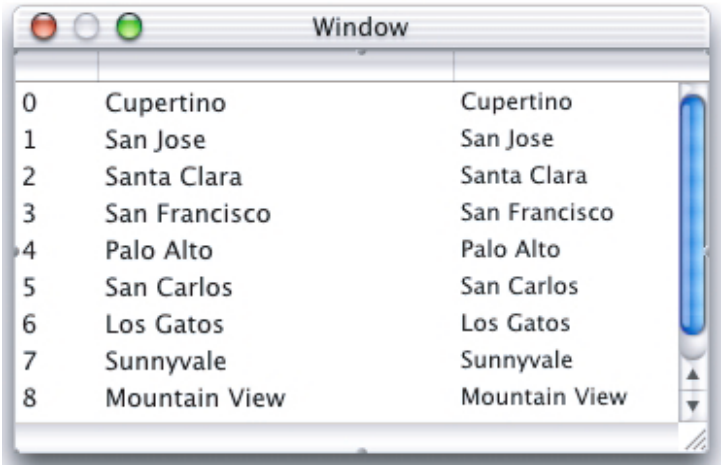

FIG. 5.5 - *Un objet table view dans Interface Builder*

Bien qu'Interface Builder fournisse des gestionnaires d'Events pour la gestion des données affichées par le tableau, l'approche conseillée et plus efficace sera d'utiliser un objet data source (page 386) pour fournir les données. La section "Exemples" de cette classe se concentre sur l'utilisation des objets table view, avec ou sans objet data source.

Propriétés des objets de la classe Table View

En plus des propriétés qu'il hérite de la classe control (page 280), un objet table view possède ces propriétés (voir la section "Version" de cette classe pour savoir dans quelle version d'AppleScript Studio sont apparues certaines propriétés) :

allows column reordering

Accès : lecture / écriture
Classe : *boolean*
Les colonnes peuvent-elles être réordonnées ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder ; si vous connectez un gestionnaire column clicked (page 430) à un objet table view, le gestionnaire ne sera pas appelé tant que la valeur de cette propriété ne vaudra pas `true`

allows column resizing

Accès : lecture / écriture
Classe : *boolean*
Les colonnes peuvent-elles être redimensionnées ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

allows column selection

Accès : lecture / écriture
Classe : *boolean*
Les colonnes peuvent-elles être sélectionnées ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

allows empty selection

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet table view autorise la sélection vide ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

allows multiple selection

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet table view autorise la sélection multiple ? Par défaut, cette propriété vaut `true` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

auto resizes all columns to fit

Accès : lecture / écriture
Classe : *boolean*
Faut-il que les colonnes soient automatiquement redimensionnées au mieux ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

auto save name

Accès : lecture / écriture
Classe : *Unicode text*
Le nom utilisé pour l'enregistrement automatique des informations sur les colonnes du tableau (voir la propriété *auto save table columns*) ; par défaut, il n'y a pas de nom ; vous pouvez régler cette propriété

dans la fenêtre Info d'Interface Builder

auto save table columns

Accès : lecture / écriture
Classe : *boolean*
Non supportée dans la version 1.3 d'AppleScript Studio ; l'ordre et la largeur des objets table column doivent-ils être automatiquement enregistrés ? Par défaut, cette propriété vaut `false` ; lorsque vous fournissez un nom à la propriété *auto save name* dans la fenêtre Info d'Interface Builder, cette propriété est automatiquement réglée sur `true`

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond de l'objet table view ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur rouge pourra être représentée par {65535, 0, 0} ; par défaut, {65535, 65535, 65535} ou la couleur blanche ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

clicked column

Accès : lecture uniquement
Classe : *integer*
L'index de la colonne ayant été cliquée pour déclencher un gestionnaire d'Events ; cette propriété vaut 0 si aucun Event ne survient ; la valeur retournée de cette méthode est significative uniquement dans les gestionnaires clicked (page 348) et double clicked (page 349)

clicked data column

Accès : lecture uniquement
Classe : *data column* (page 376)
L'objet data column ayant été cliqué ; vous permet d'obtenir directement l'objet data column cliqué et de prendre en considération ses caractéristiques de tri ; retourne rien si aucune colonne n'avait été cliquée, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

clicked data row

Accès : lecture uniquement
Classe : *data row* (page 384)

L'objet data row ayant été cliqué ; vous permet d'obtenir directement l'objet data row et de prendre en considération ses caractéristiques de tri ; retourne rien si aucune ligne n'avait été cliquée, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

clicked row

Accès : lecture uniquement
Classe : *integer*
L'index de la ligne ayant été cliquée pour déclencher un gestionnaire d'Event ; cette propriété vaut 0 si aucun Event ne survient ; la valeur retournée de cette méthode est significative uniquement dans les gestionnaires clicked (page 348) et double clicked (page 349)

corner view

Accès : lecture / écriture
Classe : *n'importe*
La view de l'angle supérieur droit (view utilisée pour dessiner la zone à droite des en-têtes des colonnes et au-dessus de l'ascenseur vertical de l'objet scroll view (page 213) ; par défaut, il s'agit d'une simple view (page 229) qui remplit simplement le cadre, mais vous pouvez la remplacer par une view personnalisée)

draws grid

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet table view dessine sa grille ? Par défaut, cette propriété vaut `false` ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

edited column

Accès : lecture uniquement
Classe : *integer*
L'index de la colonne éditée ; cette propriété vaut 0 si aucune colonne n'est éditée

edited data column

Accès : lecture uniquement
Classe : *data column* (page 376)
L'objet data column étant édité ; vous permet d'obtenir directement

l'objet data column et de prendre en considération ses caractéristiques de tri ; retourne rien si aucune colonne n'est éditée, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

edited data row

Accès : lecture uniquement
Classe : *data row* (page 384)
L'objet data row étant édité ; vous permet d'obtenir directement l'objet data row et de prendre en considération ses caractéristiques de tri ; retourne rien si aucune ligne n'est éditée, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

edited row

Accès : lecture uniquement
Classe : *integer*
L'index de la ligne étant éditée ; cette propriété vaut 0 si aucune ligne n'est éditée

grid color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de la grille ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur verte pourra être représentée par {0, 65535, 0} ; par défaut, {32767, 32767, 32767} ou la couleur grise ; vous pouvez la régler dans la fenêtre Info d'Interface Builder

header view

Accès : lecture / écriture
Classe : *table header view* (page 402)
L'objet table header view utilisé pour dessiner les en-têtes au-dessus des colonnes ; retourne rien si le tableau n'a pas d'objet table header view, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

intercell spacing

Accès : lecture / écriture
Classe : *list*
L'espace entre les cellules ; exprimé par une liste de deux nombres

row height

Accès : lecture / écriture
Classe : *real*
La hauteur de la ligne

selected column

Accès : lecture / écriture
Classe : *integer*
L'index de la colonne sélectionné ; vaut 0 si aucune colonne n'est sélectionné ; si la propriété *allows column selection* vaut `true` et *allows multiple selection* vaut `false`, vous pouvez évaluer cette propriété pour obtenir l'index de la colonne sélectionnée, s'il y en a une

selected columns

Accès : lecture / écriture
Classe : *list*
L'index de chaque colonne sélectionnée ; une liste vide si aucune colonne n'est sélectionnée ; si les propriétés *allows column selection* et *allows multiple selection* valent `true`, vous pouvez utiliser cette propriété pour déterminer les colonnes sélectionnées

selected data column

Accès : lecture / écriture
Classe : *data column* (page 376)
L'objet data column qui est sélectionné ; il n'y aura aucun objet data column sélectionné tant que la propriété *allows multiple selection* vaudra `true` ; retourne rien si aucun objet data column n'est sélectionné, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

selected data columns

Accès : lecture / écriture
Classe : *list*
Les objets data column qui sont sélectionnés ; retourne une liste vide si aucun objet data column n'est sélectionné ; si la propriété *allows mul-*

tiple selection vaut `false`, la liste retournée contiendra au maximum un seul objet data column

selected data row

Accès : lecture / écriture
Classe : *data row* (page 384)
L'objet data row qui est sélectionné ; si la propriété *allows multiple selection* vaut `false`, vous pouvez utiliser cette propriété pour obtenir l'objet data row sélectionné — autrement utilisez la propriété *selected data rows* ; retourne rien si aucun objet data row n'est sélectionné, aussi vous devrez accéder à cette valeur uniquement dans un bloc `try, on error` (pour un exemple de bloc `try, on error`, voir la section "Exemples" de la commande path for (page 112))

selected data rows

Accès : lecture / écriture
Classe : *list*
Les objets data row qui sont sélectionnés ; retourne une liste vide si aucun objet data row n'est sélectionné ; si la propriété *allows multiple selection* vaut `false`, la liste retournée contiendra au maximum un seul objet data row

selected row

Accès : lecture / écriture
Classe : *integer*
L'index de la ligne sélectionnée ; si la propriété *allows multiple selection* vaut `false`, vous pouvez vérifier cette propriété pour la ligne sélectionnée

selected rows

Accès : lecture / écriture
Classe : *list*
L'index de chaque ligne sélectionnée ; si la propriété *allows multiple selection* vaut `true`, vous pouvez vérifier cette propriété pour toutes les lignes sélectionnées

Éléments des objets de la classe Table View

En plus des éléments qu'il hérite de la classe control (page 280), un objet table view peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans

"Les formes-clés standards" (page 15).

`data source` (page 386)
spécifier par : "Les formes-clés standards" (page 15)
L'objet data source fournissant les données au tableau ; un objet table view peut avoir, soit aucun objet data source, soit un seul objet data source ; vous n'aurez pas besoin d'un numéro d'index pour vous référer à l'objet data source :

```
set theDataSource to data source of table view 1 of scroll view 1 of
  window 1
```

`table column` (page 398)
spécifier par : "Les formes-clés standards" (page 15)
Les objets table column, lesquels stockent les caractéristiques d'affichage et l'identificateur de chaque colonne

Commandes supportées par les objets de la classe Table View

Votre script peut envoyer la commande suivante à un objet table view :

`update` (page 119)

Events supportés par les objets de la classe Table View

Action

`clicked` (page 348)
`double clicked` (page 349)

Data View

`column clicked` (page 430)
`column moved` (page 430)
`column resized` (page 431)
`selection changed` (page 351)
`selection changing` (page 352)
`should select column` (page 442)
`should select row` (page 444)
`should selection change` (page 445)
`will display cell` (page 447)

Glisser-Déposer

`conclude drop` (page 481)
`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

Table View

`cell value` (page 424)
`change cell value` (page 426)
`number of rows` (page 439)

View

`bounds changed` (page 243)

Exemples

Les instructions suivantes montrent comment identifier un objet table view et lui envoyer la commande update (page 119). Les noms des objets

utilisés dans cet exemple correspondent à ceux de l'application "Table" distribuée avec AppleScript Studio.

```
set theTableView to table view "contacts" of scroll view "contacts" of window
"main"
tell theTableView to update
```

Vous pouvez utiliser des instructions, comme celles qui suivent, pour régler les lignes sélectionnées dans l'objet table view. La première instruction règle une propriété afin d'autoriser la sélection multiple dans le tableau ; la seconde sélectionne la première et la quatrième ligne de ce tableau :

```
set allows multiple selection of theTableView to true
set selected rows of theTableView to {1, 4}
```

Vous pouvez utiliser des instructions, comme celles qui suivent, pour régler les colonnes sélectionnées dans le tableau. Dans l'exemple suivant, la seconde instruction règle une propriété afin d'autoriser la sélection d'une colonne dans le tableau ; la troisième sélectionne la seconde et la troisième colonne du tableau :

```
set allows multiple selection of theTableView to true
set allows column selection of theTableView to true
set selected columns of theTableView to {2, 3}
```

Pour obtenir des informations sur une cellule d'une ligne d'un tableau avec une data source, vous pouvez utiliser les instructions suivantes du gestionnaire clicked (page 348) connecté au tableau :

```
on clicked theObject
  set rowIndex to clicked row of theObject
  if rowIndex is greater than 0 then
    set dataSource to data source of theObject
    set theRow to data row rowIndex of dataSource
    set theName to contents of data cell "name" of theRow
  end if
end clicked
```

Travailler avec les objets table view est une tâche complexe ne pouvant pas être couverte en détails ici. Pour un exemple complet, voir l'application

"Table". Cette application montre deux mécanismes pour travailler avec les objets table view. Le mécanisme recommandé, lequel fait usage d'un objet data source (page 386) pour gérer les données du tableau, est montré dans le fichier script `WithDataSource.applescript`. L'autre mécanisme moins efficace, mais pouvant tout de même convenir pour des tableaux simples, est montré dans le fichier script `WithoutDataSource.applescript`.

Pour un exemple plus détaillé, voir les chapitres du guide "*Building Applications With AppleScript Studio*" qui décrivent comment construire l'application "Mail Search" (application distribuée avec AppleScript Studio).

Discussion

Il est important que vous vous assuriez que les noms AppleScript assignés aux objets table column dans Interface Builder soient identiques à la propriété *name* de chaque objet data column de la data source de l'objet table view.

Version

Le support des Events de Glisser-Déposer est apparu dans la version 1.2 d'AppleScript Studio.

Les propriétés *edited data column*, *clicked data column*, *selected data column*, *selected data columns*, *clicked data row*, *edited data row*, *selected data row* et *selected data rows* furent ajoutées à la classe Table View (et sont donc aussi disponibles dans les sous-classes, la classe outline view (page 392)) dans la version 1.2 d'AppleScript Studio.

Ces propriétés retournent les éléments appropriés compte tenu de leurs caractéristiques de tri et devront être utilisées à la place de leurs équivalents non-triés (*edited column*, *edited row*, etc).

La propriété *auto save table columns* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Comme un bug du scripting de Cocoa a été résolu dans Mac OS X 10.2, il est dorénavant possible, depuis la version 1.2 d'AppleScript Studio, de régler une propriété qui est une liste sur une nouvelle liste. Par exemple, vous pouvez maintenant spécifier une liste pour sélectionner des lignes dans un tableau, comme il est montré dans la section "Exemples" de cette classe.

Depuis la version 1.1 d'AppleScript Studio, le comportement des objets table view et data source fut modifié comme suit : si le nom que vous réglez pour une colonne dans le champ "Identifier" du panneau "Attributes"

dans Interface Builder ne correspond pas au nom de l'objet data column (page 376) de votre data source, aucune donnée ne sera fournie à l'objet data column pour dessiner.

Toutefois, depuis la version 1.2 d'AppleScript Studio, et la version d'Interface Builder distribuée avec Mac OS X 10.2, vous pouvez nommer les colonnes d'un tableau en utilisant le champ "Name" du panneau "AppleScript" de la fenêtre Info d'Interface Builder. Utilisez un nom identificateur est encore supporté afin d'être toujours compatible avec les versions plus anciennes.

Avant la version 1.1 d'AppleScript Studio, l'application "Mail Search" s'appelait "Watson".

Chapitre 2

Commandes

Les objets basés sur les classes de la suite Data View supportent les commandes suivantes. Une **commande** est un mot ou une phrase qu'un script peut utiliser pour demander une action. Pour déterminer les commandes supportées par chaque classe, voir les descriptions propres à chacune.

append 417
item for 420

append

Ajoute la liste fournie, une liste de listes ou une liste d'enregistrements, à l'objet data source (page 386).

Les données de chaque liste ou enregistrement fournissent le contenu pour les cellules d'une seule ligne de la data source. Cette commande fournit un mécanisme simple mais très efficace pour l'ajout de données à la data source associée avec une view, comme un objet outline view (page 392) ou table view (page 403).

Si vous fournissez une liste d'enregistrements, la commande Append essaiera de faire correspondre les étiquettes de chaque enregistrement avec les identificateurs des objets data column (page 376). Pour chaque étiquette correspondant à un identificateur, elle insèrera les données de ce champ dans la colonne correspondante. Si aucune étiquette de l'enregistrement ne correspond avec l'identificateur d'une colonne, cette colonne restera vide.

Si vous fournissez une liste de listes, la commande Append fera correspondre les éléments de chaque liste avec la colonne correspondante, par

index. C'est à dire que les données du premier élément vont dans la première colonne, etc.

Syntaxe

`append`	*data source*	obligatoire
`with`	*list*	obligatoire

Paramètres

data source (page 386)
: La data source à laquelle doit venir s'ajouter les données

`with` *list*
: Une liste de listes ou une liste d'enregistrements devant venir s'ajouter à la data source spécifiée

Exemples

Le gestionnaire awake from nib (page 123) suivant est extrait de l'application "Table Sort" distribuée avec AppleScript Studio (disponible depuis la version 1.2). Ce gestionnaire, qui est connecté à l'objet table view (page 403), fait ce qui suit :

- Crée une data source nommée "names".
- Crée et ajoute quatre colonnes à la data source, une pour name, city, zip code et age. Les colonnes spécifient des préférences de classement, y compris le type et l'ordre de tri.
- Spécifie que la data source devra être triée et que la colonne courante pour le tri devra être la colonne name.
- Assigne la data source à l'objet table view (page 403) dont le gestionnaire `awake from nib` était appelé.
- Utilise la commande Append pour peupler la data source avec les données de la propriété `tableData` de l'application (montrée plus loin).

```
on awake from nib theObject
  -- Create the data source; this places it in the application
  -- object's data source elements. (Assign it to table view below.)
  set theDataSource to make new data source at end of data sources
    with properties {name:"names"}
```

```
  -- Create each of the data columns, including the sort information
  -- for each column
  make new data column at end of data columns of theDataSource
    with properties {name:"name", sort order:ascending,
    sort type:alphabetical, sort case sensitivity:case sensitive}
  make new data column at end of data columns of theDataSource
    with properties {name:"city", sort order:ascending,
    sort type:alphabetical, sort case sensitivity:case sensitive}
  make new data column at end of data columns of theDataSource
    with properties {name:"zip", sort order:ascending,
    sort type:alphabetical, sort case sensitivity:case sensitive}
  make new data column at end of data columns of theDataSource
    with properties {name:"age", sort order:ascending,
    sort type:numerical, sort case sensitivity:case sensitive}

  -- Make this a sorted data source
  set sorted of theDataSource to true

  -- Set the "name" data column as the sort column
  set sort column of theDataSource to data column "name" of theDataSource

  -- Set the data source of the table view to the new data source
  set data source of theObject to theDataSource

  -- Add the table data (using the new "append" command)
  append theDataSource with tableData
end awake from nib
```

Ci-dessous vous trouverez la propriété `tableData` définie dans l'application "Table Sort". Le champ "name" est encadré par des barres verticales afin de le différencier d'avec le mot-clé "name" du langage AppleScript :

```
property tableData : {{|name|:"Bart Simpson", city:"Springfield",
zip:"19542", age:12}, {|name|:"Ally McBeal", city:"Boston", zip:"91544",
age:28}, {|name|:"Joan of Ark", city:"Paris", zip:"53255", age:36},
{|name|:"King Tut", city:"Egypt", zip:"00245", age:45}, {|name|:"James
Taylor", city:"Atlanta", zip:"21769", age:42}}
```

Vous pouvez aussi obtenir en retour les données d'une data source (sous forme d'une liste de listes) avec la terminologie suivante (où `theDataSource` spécifie une data source) :

```
set myList to contents of every data cell of every data row of theDataSource
```

Depuis la version 1.3 d'AppleScript Studio, la commande Append accepte une simple liste d'éléments. Par exemple, vous pouvez écrire :

```
append  the data source of table view 1 of window 1 with {"One", "Two", "Three"
```

Version

Depuis la version 1.3 d'AppleScript Studio, la commande Append accepte une simple liste d'éléments.

La commande Append est apparue avec la version 1.2 d'AppleScript Studio.

item for

Retourne les éléments de la ligne choisie dans la data source.

Un objet data item (page 379) représente une seule ligne d'un objet data source (page 386).

Cette commande fonctionne uniquement avec un objet outline view (page 392).

Syntaxe

`item for`	*outline view*	obligatoire
`row`	*integer*	obligatoire

Paramètres

outline view (page 392)
: L'objet outline view à partir duquel doit être obtenu l'élément de la ligne spécifiée

`row` *integer*
: L'index de la ligne de l'objet outline view à partir de laquelle doit être obtenu l'élément

Résultats

data item (page 379)

L'objet data item de la ligne spécifiée de l'objet outline view. Retourne aucun résultat si la ligne spécifiée est hors classement

Exemples

Les instructions suivantes sont tirées du gestionnaire `mailBoxesForIndex` de l'application "Mail Search" distribuée avec AppleScript Studio. Entre autres choses, l'application "Mail Search" utilise un objet outline view (page 392) pour afficher les comptes e-mail, chacun pouvant avoir plusieurs boîtes à lettres. Une boîte à lettres, à son tour, a un nom et peut contenir des boîtes imbriquées. Le gestionnaire `mailBoxesForIndex` utilise la commande Item For pour obtenir la ligne d'un certain index, puis obtient les données (le nom de la boîte de cette ligne) à partir du premier objet data cell (page 372) de cet élément.

```
-- Determine if the selected item is an account or a mailbox
tell outline view "mailboxes" of scroll view "mailboxes"
  of split view 1 of theWindow
    set theItem to item for row mailboxIndex
    set theName to contents of data cell 1 of theItem
    -- some statements omitted
end tell
```

Chapitre 3

Events

Les objets basés sur les classes de la suite Data View supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

cell value 424
change cell value 426
change item value 427
child of item 428
column clicked 430
column moved 430
column resized 431
item expandable 432
item value 434
number of browser rows 436
number of items 437
number of rows 439
should collapse item 440
should expand item 441
should select column 442
should select item 443
should select row 444
should selection change 445
will display browser cell 446

will display cell 447
will display item cell 449
will display outline cell 450

cell value

Appelé par un objet table view (page 403) ou outline view (page 392) afin d'obtenir la valeur d'une cellule. Ce gestionnaire devra retourner la valeur de la cellule spécifiée.

La manière recommandée pour manipuler les données d'un objet table view ou outline view est d'utiliser un objet data source (page 386), dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`cell value`	*reference*	obligatoire
`row`	*integer*	obligatoire
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392) contenant la cellule

`row` *integer*
: La ligne basée sur 1 de la cellule

`table column` *table column* (page 398)
: La colonne de la cellule

Résultats

n'importe
: La valeur de la cellule de la ligne et de la colonne spécifiées. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur

Exemples

Le gestionnaire Cell Value suivant est extrait de l'application "Table" distribuée avec AppleScript Studio. Il se trouve dans le fichier script `WithoutDataSource.applescript`. L'autre manière de travailler avec les tableaux, recommandée par Apple, est présentée dans le fichier script `WithDataSource.applescript`. Ce gestionnaire :

- initialise la valeur retournée avec une chaîne vide.
- vérifie la validité du numéro de ligne.
- si le numéro existe réellement, utilise l'identificateur de la colonne pour déterminer le champ afin d'y obtenir la valeur de la cellule.
- retourne la valeur.

```
on cell value theObject row theRow table column theColumn
  -- Set the value to an empty string for now
  set theValue to ""

  -- Make sure the row we're asked for is within the number of contacts
  if (count of contacts) is greater than theRow then
    set theContact to item theRow of contacts

    -- Get the column identifier to determine which field
    -- of the record to return
    set theID to identifier of theColumn
    if theID is "name" then
      set theValue to name of theContact
    else if theID is "address" then
      set theValue to address of theContact
    else if theID is "city" then
      set theValue to city of theContact
    else if theID is "state" then
      set theValue to state of theContact
    else if theID is "zip" then
      set theValue to zip of theContact
    end if
  end if

  -- Now return the value that we set
  return theValue
```

```
end cell value
```

change cell value

Appelé par un objet table view (page 403) ou outline view (page 392) pour modifier la valeur d'une cellule.

La manière recommandée pour manipuler les données d'un objet table view ou outline view est d'utiliser un objet data source (page 386), dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`change cell value`	*reference*	obligatoire
`row`	*integer*	obligatoire
`table column`	*table column*	obligatoire
`value`	*item*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392) contenant la cellule

`row` *integer*
: La ligne de la cellule à modifier

`table column` *table column* (page 398)
: La colonne de la cellule à modifier

`value` *item* (page 60)
: La nouvelle valeur

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Change Cell Value à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra régler la cellule désignée avec la valeur spécifiée.

```
on change cell value theObject value theValue row theRow table column
tableColumn
```

```
  (*Set the specified cell to the passed value. *)
end change cell value
```

change item value

Appelé par un objet outline view (page 392) pour modifier la valeur de l'élément de la ligne spécifiée.

La manière recommandée pour manipuler les données d'un objet outline view est d'utiliser un objet data source (page 386), dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`change item value`	*reference*	obligatoire
`outline item`	*item*	obligatoire
`table column`	*table column*	obligatoire
`value`	*item*	obligatoire

Paramètres

reference
La référence de l'objet outline view (page 392) contenant l'élément

`outline item` *item* (page 60)
L'élément dont la valeur doit être modifiée

`table column` *table column* (page 398)
La colonne de l'élément à modifier

`value` *item* (page 60)
La nouvelle valeur

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Change Item Value à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra régler l'élément désigné avec la valeur spécifiée.

```
on change item value theObject value theValue outline item outlineItem table
column tableColumn
```

```
  (*Set the specified item to the passed value. *)
end change item value
```

child of item

Appelé par un objet outline view (page 392) afin d'obtenir l'élément fils spécifié d'un élément.

Bien qu'AppleScript Studio fournisse des gestionnaires pour gérer les données qu'un objet outline view affiche, comme obtenir le fils d'un élément, l'approche recommandée et plus efficace est d'utiliser un objet data source (page 386).

Syntaxe

`child of item`	*reference*	obligatoire
`outline item`	*item*	obligatoire
`[child]`	*integer*	facultatif

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant les éléments

`outline item` *item* (page 60)
: L'élément contenant l'élément fils ; généralement un numéro d'index ou une chaîne de caractères

`[child]` *integer*
: L'index du fils spécifié

Résultats

n'importe
: Le gestionnaire devra retourner le fils voulu de l'élément spécifié

Exemples

Le gestionnaire Child Of Item suivant est extrait de l'application "Outline" distribuée avec AppleScript Studio. Cette application utilise un objet outline view (page 392) pour afficher les éléments d'un système de fichiers. Ce gestionnaire :

- initialise la variable `childItem` avec une chaîne vide.

- appelle l'application Finder pour l'aider à faire ce qui suit :
  - si l'élément transmis est l'élément 0, représentant le nom du disque au plus haut niveau de l'outline, il règle `childItem` sur le nom du disque (dont l'application garde la trace à part dans la propriété `diskNames`) spécifié par le paramètre `theChild`, et le met au format string. Dans l'application "Outline", le paramètre `outline item` du gestionnaire `child of item` aura la valeur numérique 0 pour les noms de disques ; pour des éléments imbriqués, il s'agira d'un chemin délimité par deux points, comme "Hard Disk:", "Hard Disk:App Folder:", "Hard Disk:App Folder:SomeApp:", etc.
  - autrement, il règle `childItem` sur l'élément fils de l'élément transmis au format string.
    Notez que l'application "Outline" utilise le Finder pour l'aider à afficher les éléments, lesquels sont des objets que le Finder connaît bien, comme les disques, les fichiers et les dossiers. Les opérations exécutées par le Finder dans le gestionnaire sont (`get item theChild` et `get item outlineItem`), lesquelles lui demandent d'obtenir les éléments aux index spécifiés.
- retourne `childItem`.

```
on child of item theObject outline item outlineItem child theChild
  set childItem to ""

  tell application "Finder"
    if outlineItem is 0 then
      set childItem to disk (get item theChild
        of diskNames as string) as string
    else
      set childItem to item theChild of (get item outlineItem)
        as string
    end if
  end tell

  return childItem
end child of item
```

column clicked

Appelé par un objet table view (page 403) ou outline view (page 392) lorsqu'une colonne a été cliquée.

Le gestionnaire peut exécuter des opérations, comme régler les propriétés de tri des colonnes d'un objet data source (page 386).

Ce gestionnaire ne sera pas appelé tant que l'objet table view ou outline view contenant la colonne autorise le reclassement (soit que la case "Allows Reordering" du table view ou de l'outline view dans le panneau "Attributes" de la fenêtre Info d'Interface Builder est cochée, soit que la propriété *allows column property* est réglée sur `true` dans le script de l'application).

Syntaxe

`column clicked`	*reference*	obligatoire
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392)

`table column` *table column* (page 398)
: La colonne qui a été cliquée

Exemples

Pour un exemple de gestionnaire Column Clicked, voir la section "Exemples" de la classe data source (page 386).

column moved

Appelé par un objet table view (page 403) ou outline view (page 392) après qu'une colonne ait été déplacée, comme lorsque l'utilisateur mélange les colonnes.

Syntaxe

`column moved`	*reference*	obligatoire
`new column`	*integer*	obligatoire
`old column`	*integer*	obligatoire

Paramètres

reference
La référence de l'objet table view (page 403) ou outline view (page 392)

`new column` *integer*
L'index de la nouvelle position de la colonne

`old column` *integer*
L'index de l'ancienne position de la colonne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Column Moved à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Les paramètres fournissent les index de l'ancienne position et de la nouvelle position de la colonne.

```
on column moved theObject new column newColumn old column oldColumn
  (* Respond to changed column position. *)
end column moved
```

column resized

Appelé par un objet table view (page 403) ou outline view (page 392) après qu'une colonne soit redimensionnée. La nouvelle taille peut être la même que l'ancienne.

Syntaxe

`column resized`	*reference*	obligatoire
`old width`	*real*	obligatoire
`table column`	*table column*	obligatoire

Paramètres

reference
La référence de l'objet table view (page 403) ou outline view (page 392)

`old width` *real*
L'ancienne largeur de la colonne

`table column` *table column* (page 398)
La colonne pouvant avoir été redimensionnée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Column Resized à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra exécuter toute opération requise par la modification de la taille de la colonne, peut-être après avoir d'abord déterminé si la largeur de la colonne a réellement été modifiée.

```
on column resized theObject table column tableColumn old width oldWidth
  if width of tableColumn is not equal to oldWidth then
    (* Add statements to perform any operations required after change
      of column width. *)
  end if
end column resized
```

item expandable

Appelé par un objet outline view (page 392) pour découvrir si l'élément spécifié est développé.

Le gestionnaire retournera `true` si l'élément est développé, `false` dans le cas contraire.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet outline view, dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`item expandable`	*reference*	obligatoire
`[outline item]`	*item*	facultatif

Paramètres

reference
La référence de l'objet outline view (page 392) contenant l'élément

`[outline item]` *item* (page 60)
L'élément pouvant être développé

Résultats

boolean
Retourne `false` si l'élément n'est pas développé ou `true` s'il l'est. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Le gestionnaire Item Expandable suivant est extrait de l'application "Outline" distribuée avec AppleScript Studio. Cette application utilise un objet outline view (page 392) pour afficher les éléments d'un système de fichiers. Ce gestionnaire :

- règle la variable `isExpandable` sur `false`.
- si l'élément transmis est l'élément O, représentant les noms des disques au plus haut niveau de l'outline, et s'il y a plusieurs noms de disque (dont l'application garde la trace à part dans la propriété `diskNames`), il règle `isExpandable` sur `true`.
- autrement, il appelle l'application Finder pour obtenir le nombre total d'éléments de l'élément. Si ce total est supérieur à 1, il règle `isExpandable` sur `true`.
- retourne `isExpandable`.

```
on item expandable theObject outline item outlineItem
  set isExpandable to false

  if outlineItem is 0 then
    if (count of diskNames) is greater than 1 then
      set isExpandable to true
    end if
  else
    tell application "Finder"
      if (count of items of (get item outlineItem))
        is greater than 1 then
          set isExpandable to true
      end if
    end tell
```

```
    end if

    return isExpandable
end item expandable
```

item value

Appelé par un objet outline view (page 392) pour obtenir la valeur d'un élément.

Le gestionnaire retournera la valeur (généralement sous la forme d'une chaîne de caractères) devant être affichée de l'élément spécifié.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet outline view, dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Pour les objets outline view qui n'utilisent pas une data source, ce gestionnaire est appelé une fois pour chaque ligne affichée dans chaque colonne. Par exemple, si l'objet outline view affiche la hiérarchie d'un système de fichiers dans trois colonnes, une pour le nom, une autre pour la date de modification et une autre pour la taille de chaque élément, l'objet outline view appellera le gestionnaire Item Value trois fois pour chaque élément affiché (une première fois pour le nom, une seconde fois pour la date de modification et une troisième fois pour la taille).

Ce gestionnaire n'est pas appelé pour les lignes développées non visibles.

Syntaxe

`item value`	*reference*	obligatoire
`[outline item]`	*item*	facultatif
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant l'élément

`[outline item]` *item* (page 60)
: L'élément dont la valeur doit être obtenue

`table column` *table column* (page 398)
: La colonne de l'élément

Résultats

n'importe
: La valeur de l'élément spécifié ; généralement retournée au format string

Exemples

Le gestionnaire Item Value suivant est extrait de l'application "Outline" distribuée avec AppleScript Studio. Cette application utilise un objet outline view (page 392) pour afficher les éléments d'un système de fichiers. Ce gestionnaire utilise l'identificateur de la colonne pour déterminer le type de valeur à retourner pour l'élément. Il appelle alors l'application Finder pour obtenir la valeur (soit le nom, la date de modification ou le type) et retourne la valeur au format string.

```
on item value theObject outline item theItem table column theColumn
  set itemValue to ""

  if the identifier of theColumn is "name" then
    tell application "Finder"
      set itemValue to displayed name of (get item theItem)
        as string
    end tell
  else if the identifier of theColumn is "date" then
    tell application "Finder"
      set itemValue to modification date of
        (get item theItem) as string
    end tell
  else if the identifier of theColumn is "kind" then
    tell application "Finder"
      set itemValue to kind of (get item theItem) as string
    end tell
  end if

  return itemValue
end item value
```

number of browser rows

Appelé pour obtenir le nombre de lignes d'un objet browser (page 363) pour une colonne donnée.

Contrairement aux autres data views comme outline view (page 392) et table view (page 403), vous ne pourrez pas alimenter en données un objet browser avec un objet data source (page 386). En conséquence de quoi, les performances pourront être insuffisantes pour des objets browser affichant plus qu'un petit nombre d'éléments, aussi vous devrez préférer l'utilisation d'une des deux autres data views, bien sûr si cela est compatible avec vos impératifs.

Syntaxe

`number of browser rows`	*reference*	obligatoire
`in column`	*integer*	obligatoire

Paramètres

reference
: La référence de l'objet browser (page 363) duquel doit être obtenu le nombre de lignes

`in column` *integer*
: L'index de la colonne

Résultats

integer
: Retourne le nombre de lignes de la colonne spécifiée de l'objet browser

Exemples

Le gestionnaire Number Of Browser Rows suivant est extrait de l'application "Browser" distribuée avec AppleScript Studio. Cette application navigue dans un système de fichiers, affichant les fichiers et les dossiers dans une fenêtre identique au mode colonne de l'application Finder. Ce gestionnaire :

- règle la variable `rowCount` sur 0.
- s'il n'y a aucun nom de disque affiché dans la première colonne de l'objet browser, il ne modifie pas la valeur de `rowCount`.

- s'il y a au moins un nom de disque affiché dans la première colonne (`(count of diskNames) > 0`), et si la colonne spécifiée est la première, il règle `rowCount` sur le nombre total de noms de disque (dont l'application garde la trace à part dans la propriété `diskNames`).
- s'il y a au moins un nom de disque mais la colonne spécifiée n'est pas la première, il obtient le chemin de la colonne, puis appelle l'application Finder pour obtenir le nombre total d'éléments de ce chemin. Puis il règle `rowCount` sur le total retourné par le Finder.
- retourne `rowCount`.

```
on number of browser rows theObject in column theColumn
  set rowCount to 0

  if (count of diskNames) > 0 then
    if theColumn is 1 then
      set rowCount to count of diskNames
    else
      tell browser "browser" of window "main"
        set thePath to path for column theColumn - 1
      end tell

      tell application "Finder"
        set rowCount to count of items of item thePath
      end tell
    end if
  end if

  return rowCount
end number of browser rows
```

number of items

Appelé par un objet outline view (page 392) pour obtenir le nombre d'éléments fils de l'élément spécifié.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet outline view, dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`number of items`	*reference*	obligatoire
`[outline item]`	*item*	facultatif

Paramètres

reference
La référence de l'objet outline view (page 392) contenant les éléments

`[outline item]` *item* (page 60)
L'élément duquel doit être obtenu le nombre d'éléments contenus

Résultats

integer
Retourne le nombre d'éléments contenus dans l'élément spécifié de l'objet outline view

Exemples

Le gestionnaire Number Of Items suivant est extrait de l'application "Outline" distribuée avec AppleScript Studio. Cette application utilise un objet outline view (page 392) pour afficher les éléments d'un système de fichiers. Ce gestionnaire utilise l'application Finder pour compter le nombre d'éléments de l'élément spécifié. Ce processus est décrit plus en détails dans les sections "Exemples" des gestionnaires item expandable (page 432) et number of browser rows (page 436).

```
on number of items theObject outline item outlineItem
  set itemCount to 0

  tell application "Finder"
    if (count of diskNames) > 0 then
      if outlineItem is 0 then
        set itemCount to count of diskNames
      else
        set itemCount to count of items of (get item outlineItem)
      end if
    end if
  end tell

  return itemCount
```

```
end number of items
```

number of rows

Appelé par un objet table view (page 403) ou outline view (page 392) pour obtenir le nombre de lignes.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet table view ou outline view, dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`number of rows` *reference* obligatoire

Paramètres

reference

La référence de l'objet table view (page 403) ou outline view (page 392) duquel doit être obtenu le nombre de lignes

Résultats

integer

Retourne le nombre de lignes de l'objet table view ou outline view

Exemples

Le gestionnaire Number Of Rows suivant est extrait de l'application "Table" distribuée avec AppleScript Studio. Vous le trouverez dans le fichier script `withoutDataSource.applescript`. Cette application montre comment travailler avec les données affichées dans les lignes et les colonnes d'un objet table view (page 403).

Notez, toutefois, que bien que vous puissiez travailler avec les données d'un table view sans un objet data source (page 386), comme dans cet exemple, l'approche conseillée et plus efficace est d'utiliser une data source. L'application "Table" montre cette approche, laquelle ne requiert pas de gestionnaire Number Of Rows, dans le fichier script `withDataSource.applescript`.

Le gestionnaire suivant retourne juste le nombre total de contacts.

```
on number of rows theObject
```

```
  return count of contacts
end number of rows
```

should collapse item

Appelé par un objet outline view (page 392) pour déterminer si un élément devra être réduit.

Le gestionnaire retournera `true` pour autoriser la réduction de l'élément, `false` pour l'interdire.

Syntaxe

`should collapse item`	*reference*	obligatoire
`[outline item]`	*item*	facultatif

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant les éléments pouvant être réduits

`[outline item]` *item* (page 60)
: L'élément pouvant être réduit

Résultats

boolean
: Retourne `true` pour autoriser l'élément à être réduit ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Collapse Item à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre `theObject` référence l'objet outline view. Votre gestionnaire devra déterminer s'il autorise l'élément spécifié à être réduit, puis retournera la valeur appropriée.

```
on should collapse item theObject outline item outlineItem
```

```
  set allowCollapse to false
  --Check variable, perform test, or call handler to see if OK to collapse
  -- If so, set allowCollapse to true
  return allowCollapse
end should collapse item
```

should expand item

Appelé par un objet outline view (page 392) pour déterminer si un élément doit être développé.

Ce gestionnaire retournera `true` pour autoriser le développement de l'élément, `false` pour l'empêcher.

Syntaxe

`should expand item`	*reference*	obligatoire
`[outline item]`	*item*	facultatif

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant les éléments pouvant être développés

`[outline item]` *item* (page 60)
: L'élément pouvant être développé

Résultats

boolean
: Retourne `true` pour autoriser l'élément à être développé ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Expand Item à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre `theObject` référence l'objet outline view. Votre gestionnaire devra déterminer s'il autorise l'élément spécifié à être développé,

puis retournera la valeur appropriée.

```
on should expand item theObject outline item outlineItem
  set allowExpand to false
  --Check variable, perform test, or call handler to see if OK to expand
  -- If so, set allowExpand to true
  return allowExpand
end should expand item
```

should select column

Appelé pour déterminer si la sélection est autorisée lorsque l'utilisateur clique sur une colonne dans un objet table view (page 403) ou outline view (page 392) (comme lorsque l'utilisateur essaie de faire glisser une colonne pour modifier sa position).

Ce gestionnaire devra retourner `true` pour autoriser la sélection ou `false` pour la refuser.

Par défaut, la sélection d'une colonne est activée pour les objets table view (page 403) mais pas pour les objets outline view (page 392), mais vous pouvez modifier ce réglage dans Interface Builder. Vous n'aurez pas besoin de connecter ce gestionnaire à moins que vous souhaitiez autoriser la sélection d'une colonne dans certains cas mais l'interdire dans d'autres.

Syntaxe

`should select column`	*reference*	obligatoire
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392) contenant la colonne

`table column` *table column* (page 398)
: La colonne devant être sélectionnée

Résultats

boolean

Retourne `true` pour autoriser la colonne à être sélectionnée ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Select Column à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il autorise la colonne spécifiée à être sélectionnée, puis retournera la valeur appropriée.

```
on should select column theObject table column tableColumn
  set allowSelection to false
  --Check variable, perform test, or call handler to see if OK to select
  -- If so, set allowSelection to true
  return allowSelection
end should select column
```

should select item

Appelé pour déterminer si la sélection est autorisée lorsque l'utilisateur clique sur un élément dans un objet outline view (page 392).

Ce gestionnaire devra retourner `true` pour autoriser la sélection ou `false` pour la refuser. Vous n'aurez pas besoin de connecter ce gestionnaire à moins que vous souhaitiez autoriser la sélection d'un élément dans certains cas mais l'interdire dans d'autres.

Syntaxe

`should select item`	*reference*	obligatoire
`[outline item]`	*item*	facultatif

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant l'élément pouvant être sélectionné

`outline item` *item* (page 60)
: L'élément pouvant être sélectionné

Résultats

boolean
: Retourne `true` pour autoriser l'élément à être sélectionné ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Select Item à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il autorise l'élément spécifié à être sélectionné, puis retournera la valeur appropriée.

```
on should select item theObject outline item outlineItem
  set allowSelection to false
  --Check variable, perform test, or call handler to see if OK to select
  -- If so, set allowSelection to true
  return allowSelection
end should select item
```

should select row

Appelé pour déterminer si la sélection est autorisée lorsque l'utilisateur clique sur une ligne dans un objet table view (page 403) ou outline view (page 392).

Ce gestionnaire devra retourner `true` pour autoriser la sélection ou `false` pour la refuser. Vous n'aurez pas besoin de connecter ce gestionnaire à moins que vous souhaitiez autoriser la sélection d'une ligne dans certains cas mais l'interdire dans d'autres.

Syntaxe

`should select row`	*reference*	obligatoire
`row`	*integer*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392) contenant la ligne pouvant être sélectionnée

`row` *integer*
: L'index de la ligne devant être sélectionnée

Résultats

boolean
: Retourne `true` pour autoriser la ligne à être sélectionné ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Select Row à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il autorise la ligne spécifiée à être sélectionnée, puis retournera la valeur appropriée.

```
on should select row theObject row theRow
  set allowSelection to false
  --Check variable, perform test, or call handler to see if OK to select
  -- If so, set allowSelection to true
  return allowSelection
end should select row
```

should selection change

Appelé par un objet table view (page 403) ou outline view (page 392) pour déterminer si la sélection courante devra être modifiée.

Ce gestionnaire retournera `true` pour autoriser la modification de la sélection, `false` pour la refuser.

Syntaxe

`should selection change` *reference* obligatoire

Paramètres

reference

La référence de l'objet table view (page 403) ou outline view (page 392) pour lequel la sélection peut être modifiée

Résultats

boolean

Retourne `true` pour autoriser la modification de la sélection ; `false` pour l'empêcher. Si vous implémentez ce gestionnaire, vous devrez obligatoirement retourner une valeur booléenne

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Should Selection Change à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra déterminer s'il autorise la modification de la sélection, puis retournera la valeur appropriée.

```
on should selection change theObject
  set allowSelectionChange to false
  --Check variable, perform test, or call handler to see if OK to select
  -- If so, set allowSelectionChange to true
  return allowSelectionChange
end should selection change
```

will display browser cell

Appelé avant qu'un objet browser cell (page 370) ne soit affiché dans l'objet browser (page 363).

Ce gestionnaire ne pourra pas annuler l'opération d'affichage, mais peut la préparer.

Contrairement aux autres data views comme outline view (page 392) et table view (page 403), vous ne pourrez pas alimenter en données un objet browser avec un objet data source (page 386). En conséquence de quoi, les performances pourront être insuffisantes pour des objets browser affichant plus qu'un petit nombre d'éléments, aussi vous devrez préférer l'utilisation d'une des deux autres data views, bien sûr si cela est compatible avec vos impératifs.

Syntaxe

`will display browser cell`	*reference*	obligatoire
`browser cell`	*browser cell*	obligatoire
`in column`	*integer*	obligatoire
`row`	*integer*	obligatoire

Paramètres

reference
: La référence de l'objet browser (page 363) contenant la cellule devant être affichée

`browser cell` *browser cell* (page 370)
: La cellule devant être affichée

`in column` *integer*
: L'index de la colonne de la cellule donnée

`row` *integer*
: L'index de la ligne de la cellule donnée

Exemples

Pour un exemple de gestionnaire Will Display Browser Cell, voir la section "Exemples" de la classe browser cell (page 370).

`will display cell`

Appelé avant qu'une cellule d'un objet table view (page 403) ou outline view (page 392) ne soit affichée (par conséquent afficher la donnée dans la ligne et la colonne spécifiées).

Ce gestionnaire ne pourra pas annuler l'opération d'affichage, mais peut la préparer.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet table view ou outline view, dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`will display cell`	*reference*	obligatoire
`cell`	*n'importe*	obligatoire
`row`	*integer*	obligatoire
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet table view (page 403) ou outline view (page 392) contenant la cellule devant être affichée

`cell` *n'importe*
: La cellule qui est sur le point d'être affichée ; voir cell (page 264), image cell (page 284) et text field cell (page 330)

`row` *integer*
: L'index de la ligne de la cellule devant être affichée

`table column` *table column* (page 398)
: La colonne de la cellule devant être affichée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Display Cell à un objet table view (page 403) ou outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra faire n'importe quelle préparation requise par l'affichage de la cellule spécifiée.

```
on will display cell theObject row theRow cell theCell table column
tableColumn
  (* Prepare for cell to be displayed. *)
end will display cell
```

`will display item cell`

Appelé avant qu'une donnée soit affichée dans un objet outline view (page 392) (par conséquent afficher la donnée de l'élément et de la colonne spécifiés).

Ce gestionnaire ne pourra pas annuler l'opération d'affichage, mais peut la préparer.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet outline view , dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`will display item cell`	*reference*	obligatoire
`cell`	*n'importe*	obligatoire
`[outline item]`	*item*	facultatif
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant l'élément devant être affiché

`cell` *n'importe*
: La cellule qui est sur le point d'être affichée ; voir cell (page 264), image cell (page 284) et text field cell (page 330)

`[outline item]` *item* (page 60)
: L'élément contenant la cellule qui sera affichée

`table column` *table column* (page 398)
: La colonne de la cellule devant être affichée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Display Item Cell à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Votre gestionnaire devra faire n'importe quelle préparation requise par l'affichage de la cellule de l'élément spécifié.

```
on will display item cell theObject outline item outlineItem cell theCell
```

```
table column tableColumn
  (* Prepare for cell to be displayed. *)
end will display item cell
```

will display outline cell

Appelé avant que la cellule implémentant le triangle de développement ne soit affichée dans l'objet outline view (page 392).

Ce gestionnaire ne pourra pas annuler l'opération d'affichage, mais peut la préparer.

Apple recommande d'utiliser un objet data source (page 386) pour fournir en données un objet outline view , dans ce cas ce gestionnaire n'est pas utile (ou appelé).

Syntaxe

`will display outline cell`	*reference*	obligatoire
`cell`	*n'importe*	obligatoire
`[outline item]`	*item*	facultatif
`table column`	*table column*	obligatoire

Paramètres

reference
: La référence de l'objet outline view (page 392) contenant l'outline cell qui sera affiché

`cell` *n'importe*
: La cellule qui est sur le point d'être affichée ; voir cell (page 264), image cell (page 284) et text field cell (page 330)

`[outline item]` *item* (page 60)
: L'élément contenant la cellule qui sera affichée

`table column` *table column* (page 398)
: La colonne de la cellule devant être affichée

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Will Display Outline Cell à un objet outline view (page 392), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique

à celui qui suit. Le paramètre `theObject` référence l'objet outline view (page 392). Votre gestionnaire devra faire n'importe quelle préparation requise par l'affichage de l'outline cell spécifié.

```
on will display outline cell theObject outline item outlineItem cell theCell
table column tableColumn
  (* Prepare for cell to be displayed. *)
end will display outline cell
```

Sixième partie

Document Suite

Cette partie décrit la terminologie de la suite Document d'AppleScript Studio.

La suite Document fournit la terminologie pour le travail avec les documents dans les applications AppleScript Studio. Cette suite définit la version AppleScript Studio de la classe document (page 457), laquelle remplace celle définie dans la Standard Suite de Cocoa (décrite dans la section "La terminologie fournie par le framework Cocoa Application" (page 13)).

La suite Document est apparue avec la version 1.2 d'AppleScript Studio afin de faciliter la création des applications basées sur le modèle "AppleScript Document-based Application". Cette suite définit deux gestionnaires d'Events de haut niveau (data representation (page 465) et load data representation (page 467)), ainsi que deux gestionnaires de bas niveau (read from file (page 469) et write to file (page 470)). De plus, la classe window (page 76) possède désormais un élément `document` donnant accès à son ou ses documents depuis l'interface utilisateur.

Xcode fournit le modèle de projet "AppleScript Document-based Application" pour les applications qui créent et gèrent de multiples document. Pour supporter la suite Document, les réglages de ce modèle de projet furent révisés. Le fichier script `Document.applescript` du projet intègre dorénavant des versions vierges des gestionnaires data representation (page 465) et load data representation (page 467). L'application "Task List", disponible depuis la version 1.2 d'AppleScript Studio, montre comment lire et écrire des fichiers simples avec ces gestionnaires de haut niveau. L'application "Plain Text Editor", disponible aussi depuis la version 1.2, montre comment lire et écrire des fichiers texte légèrement plus complexes avec les gestionnaires de bas-niveau (read from file (page 469) et write to file

(page 470)).

Quelque soit les gestionnaires utilisés, le grand avantage du support des documents par AppleScript Studio est que vos applications n'auront plus besoin de construire des panels d'ouverture, d'enregistrement ou d'enregistrement sous, ou même de vous soucier des noms de fichier choisis par l'utilisateur. L'application ne fera que lire ou écrire les données lorsque le gestionnaire approprié est appelé. Pour les gestionnaires de bas niveau, elle utilisera le nom du fichier transmis au gestionnaire.

Les classes et Events de la suite Document sont décrits dans les chapitres suivants :

Classes 457
Events 465

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Document contient la classe suivante :

document 457

`document`

Pluriel : `documents`
Hérite de : `responder` (page 68)
Classe Cocoa : NSDocument

Représente les données affichées dans les fenêtres pouvant généralement être lues depuis les fichiers ou écrites dedans.

Le framework Cocoa application fournit un large support au document de base et, depuis la version 1.2 d'AppleScript Studio, vous pouvez tirer partie de ce support dans les applications AppleScript Studio basées sur le modèle "AppleScript Document-based Application". Par exemple, si vous créez un nouveau projet "AppleScript Document-based Application" dans Xcode, sans effectuer de modifications, l'application pourra ouvrir de multiples fenêtres "Sans-titre" et même les enregistrer, quoique sans données application.

L'illustration 6.1 montre le panneau "Groups & Files" d'un nouveau projet "Document-based" (nommé "DefaultDocumentProject"), avec plusieurs groupes développés. La plupart de ces éléments sont communs à tous les modèles de projet AppleScript Studio, et sont décrits dans le guide "*Building Applications With AppleScript Studio*" (voir "Autres documentations"

(page 5) pour plus d'informations sur ce guide). Ce qui suit est une description des éléments par défaut qui sont propres à un projet "AppleScript Document-based Application" :

- `Document.applescript` est le ficher script par défaut d'un document. Depuis la version 1.2 d'AppleScript Studio, ce fichier contient des gestionnaires data representation (page 465) et load data representation (page 467) vierges. Vous devrez remplir ces gestionnaires pour fournir les données pour l'enregistrement du document et pour charger les données lues depuis.
  Si vous avez d'autres instructions, vous pouvez les ajouter à ce fichier.
- `Document.nib` est le fichier Nib pour la création de fenêtres document-associé. Les fichiers Nib sont décrits avec le gestionnaire awake from nib (page 123).
- `Credits.rtf` est un fichier au format "Rich Text Format" qui fournit le texte pour la fenêtre par défaut "À propos de..." d'une application Cocoa. Vous éditerez ce fichier pour pouvoir y inscrire vos propres informations.

Le framework `Cocoa.framework` est listé dans le groupe "Linked Frameworks". Sa case "Target" est cochée, indiquant qu'il fait partie de la cible courante. Le groupe "Headers" de Cocoa contient actuellement un seul fichier en-tête, `Cocoa.h` (non visible). Ce fichier importe les fichiers en-tête `Foundation.h` et `AppKit.h`, deux fichiers qui à leur tour importent tous les fichiers en-têtes pour les deux frameworks composant Cocoa, `AppKit.framework` et `Foundation.framework`.

Vous pouvez voir dans l'illustration 6.1 que le groupe "Other Frameworks" contient à la fois `AppKit.framework` et `Foundation.framework` (cela est vrai pour tous les projets AppleScript Studio). Ces frameworks sont listés dans le groupe "Other Frameworks" pour fournir un accès pratique et aisé aux fichiers en-têtes contenant toutes les classes Cocoa, méthodes et constantes disponibles pour vos applications AppleScript Studio. Vos applications peuvent faire usage de ces informations dans du code Objective-C ou autre que vous écrirez, ou dans les scripts utilisant la commande call method (page 94).

Les dossiers nommés `Resources/English.lprog/Documentation` dans les frameworks AppKit et Foundation contiennent la documentation Cocoa. Vous pourrez accéder à cette même documentation en choisissant "Cocoa Help" dans le menu "Help" de Xcode.

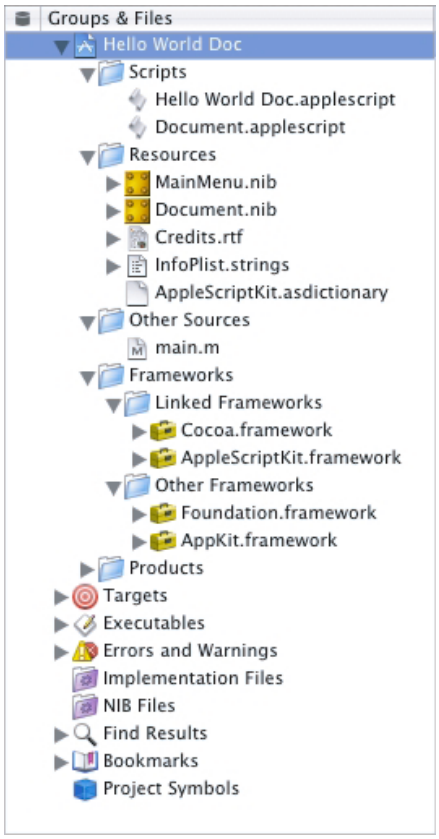

FIG. 6.1 - *Le panneau "Groups & Files" d'un projet Xcode*

Après création d'un projet "Document-based Application" par défaut, vous aurez encore quelques travaux importants à faire pour tirer pleinement avantage du support des documents par AppleScript Studio. Ce travail consistera à :

- Ajouter des éléments d'interface au fichier Nib du document.
- Fournir les données pour l'écriture du fichier et les extraire lors de sa lecture :
  - Pour des documents simples, vous pouvez utiliser les gestionnaires de haut niveau, data representation (page 465) et load data representation (page 467).
  - Pour des documents plus complexes, vous pourrez utiliser à la place les gestionnaires de bas niveau, write to file (page 470) et read from file (page 469).
- Fournir des informations sur le type de document dans Xcode. Vous

pouvez utiliser le “Target editor” (vous y accéderez en choisissant le menu “Edit Active Target ‘nomAppli’” du menu “Project” ou en appuyant sur les touches Option + Cmd + E) pour spécifier :
- le nom du type de document
- les extensions associées (comme `"txt"`)
- les types d'OS associés (un code sur quatre caractères, comme `"TEXT"`)
- si l'application peut éditer le type de fichier, ou juste le visualiser
- l'icone associé avec le type de document

La version minimale pour la lecture et l'écriture des fichiers sera de fournir le type de document, son extension et spécifier si l'application peut éditer ce type.

Que vous utilisiez les gestionnaires de haut niveau ou de bas niveau pour lire ou écrire des données, le grand avantage du support des documents par AppleScript Studio est que vos applications n'auront plus besoin de construire des panels d'ouverture, d'enregistrement ou d'enregistrement sous, ou même de vous soucier des noms de fichier choisis par l'utilisateur. L'application ne fera que lire ou écrire les données lorsque le gestionnaire approprié est appelé. Pour les gestionnaires de bas niveau, elle utilisera le nom du fichier transmis au gestionnaire.

AppleScript Studio inclut deux exemples d'application, disponible depuis la version 1.2, qui montrent comment travailler avec les applications basées sur le modèle “AppleScript Document-based Application”. Voir la section “Exemples” pour plus d'informations.

Propriétés des objets de la classe Document

En plus des propriétés qu'il hérite de la classe responder (page 68), un objet document possède ces propriétés :

file name

Accès : lecture / écriture
Classe : *Unicode text*
Le nom de fichier du document ; non défini pour un nouveau document jusqu'à ce qu'il soit réglé (comme lors de l'enregistrement) ; une chaîne de caractères au format POSIX (délimité par des slashs) ; par exemple, `/Users/votreUtilisateur/Documents/fichier.txt`.

file kind

Accès : lecture / écriture
Classe : *Unicode text*
Synonyme de la propriété *file type* ; ajoutée dans la version 1.3 d'AppleScript Studio uniquement pour une rétro-compatibilité — n'utilisez pas cette propriété dans vos scripts ; voir la section "Version" pour plus d'informations

file type

Accès : lecture / écriture
Classe : *Unicode text*
Le type de fichier du document ; il ne s'agit pas du code sur quatre caractères que vous pouviez connaître dans Mac OS Classique ; il s'agit d'une chaîne de caractères, comme "DocumentType", que vous règlerez dans Xcode (vous pouvez aussi dans Xcode définir le type sur quatre caractères, ainsi que l'extension, pour les documents) ; voir la section "Version" pour plus d'informations

modified

Accès : lecture / écriture
Classe : *boolean*
Le document a-t-il été modifié ?

name

Accès : lecture uniquement
Classe : *Unicode text*
Le nom du document ; par défaut, "Sans-titre" pour le premier document ; l'unique façon de modifier le nom est d'utiliser "Enregistrer sous..." pour enregistrer le document avec un nouveau nom, ou de le modifier séparément sur le disque

path

Accès : lecture / écriture
Classe : *Unicode text*
Le chemin du document (y compris son nom de fichier) ; ajoutée dans la version 1.3 d'AppleScript Studio ; non définie pour un nouveau document jusqu'à ce qu'il soit enregistré ; au format POSIX, une chaîne de caractères délimitées par des slashs ; par exemple `/users/yourUser/Documents/fichier.text`

Éléments des objets de la classe Document

Un objet document peut contenir les éléments listés ci-dessous. Votre script peut, pour la plupart, les spécifier avec les formes-clés décrites dans "Les formes-clés standards" (page 15). Consulter la section "Version" de cette classe pour connaître dans quelle version d'AppleScript Studio un élément a pu être ajouté.

`window` (page 76)
Spécifier par : "Les formes-clés standards" (page 15)
Les fenêtres du document ; voir la section "Version" ci-dessous

Commandes supportées par les objets de la classe Document

`close` (de la Core Suite Cocoa)
`print` (de la Core Suite Cocoa)
`save` (de la Core Suite Cocoa)

Events supportés par les objets de la classe Document

Un objet document supporte les gestionnaires répondant aux Events suivants :

Document

`data representation` (page 465)
`load data representation` (page 467)
`read from file` (page 469)
`write to file` (page 470)

Nib

`awake from nib` (page 123)

Exemples

L'application "Plain Text Editor", distribuée avec AppleScript Studio depuis la version 1.2, montre comment lire et écrire des fichiers texte avec les gestionnaires de bas niveau read from file (page 469) et write to file (page 470). Les sections "Exemples" de ces gestionnaires montrent les versions de ces gestionnaires dans l'application "Plain Text Editor". L'application "Task List", également distribuée depuis la version 1.2 d'AppleScript

Studio, montre comment lire et écrire des fichiers texte avec les gestionnaires de haut niveau data representation (page 465) et load data representation (page 467). Les sections "Exemples" de ces gestionnaires montrent les versions de ces gestionnaires dans l'application "Task List".

Vous pouvez utiliser les instructions suivantes dans l'application "Éditeur de Scripts" pour accéder aux propriétés d'un document d'une application basée sur le modèle "AppleScript Document-based Application". Ces mêmes instructions fonctionneront dans le script d'une application AppleScript Studio (bien que vous n'aurez pas besoin de les encadrer par un bloc `tell application`).

```
tell application "Document Application"
  set myName to name of the first document
  -- result: "Untitled 2"
end
```

Version

La propriété *path* fut ajoutée dans la version 1.3 d'AppleScript Studio.

Avant la version 1.3 d'AppleScript Studio, la propriété *file type* était implémentée avec un code sur 4 caractères pouvant provoquer des conflits de terminologie avec d'autres applications. Depuis la version 1.3, cette propriété est implémentée avec un code différent évitant ces conflits possibles. La propriété *file kind* fut ajoutée pour une rétro-compatibilité avec des scripts existant, mais elle ne devra pas être utilisée dans vos scripts.

Les changments suivants furent faits dans la version 1.2 d'AppleScript Studio :

- La classe Document fut déplacée de la suite Application vers sa propre suite Document.
- La propriété *file type* fut ajoutée.
- Le support des Events suivants fut ajouté : data representation (page 465), load data representation (page 467), read from file (page 469) et write to file (page 470).
- Vous ne pourrez pas connecter un gestionnaire will open (page 165) à un document. Toutefois, vous pourrez connecter ce gestionnaire à la fenêtre du document (voir le prochain élément).
- L'élément window (page 76) fut ajouté à la classe Document afin de fournir l'accès aux éléments d'interface associés avec le document.

Notez que comme `window` est un élément (adressable par le nom, le numéro d'index, ID, etc) et non une propriété, l'instruction `window of document 1` retournera une liste telle que {`window id 1`}). De plus, `window` est synonyme de `windows` pour un élément.

- Notez aussi que l'élément `document` fut ajouté à la classe window (page 76), pour que les éléments d'interface des fenêtres ayant un document associé puissent y accéder.

Chapitre 2

Events

Les objets basés sur les classes de la suite Document supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Vous utiliserez ces gestionnaires avec les objets de la classe document (page 457). Pour déterminer les Events supportés par une classe, se référer à sa description.

data representation 465
load data representation 467
read from file 469
write to file 470

`data representation`

Appelé losqu'un document est sur le point d'être enregistré pour incorporer les données au document.

Ce gestionnaire est appelé comme le résultat de l'ouverture par l'utilisateur des panels d'enregistrement ("Enregistrer" et "Enregistrer sous...") (ou en utilisant les raccourcis-claviers équivalents) et du choix d'enregistrer le document.

Il s'agit d'un gestionnaire de haut niveau que vous pouvez utiliser lorsque vous voulez créer des documents qui seront spécifiques à votre application. Ce gestionnaire retournera alors les données du document sous la forme que vous choisirez, comme une simple chaîne de caractères, une liste, un enregistrement ou un autre type de données. L'application ne devra pas

traiter avec l'ouverture du fichier et l'écriture des données — AppleScript Studio enregistrera automatiquement les données dans le document.

Le contraire de Data Representation est load data representation (page 467).

Syntaxe

`data representation`	*reference*	obligatoire
`of type`	*Unicode text*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire Data Representation est appelé

`of type` *Unicode text*
: Le type (extension) du fichier document

Exemples

L'application "Task List" distribuée depuis la version 1.2 d'AppleScript Studio, fournit le gestionnaire suivant pour montrer le mécanisme de haut niveau permettant l'écriture des données dans les fichiers. Dans ce cas précis, le gestionnaire obtient des informations de l'objet data source (page 386) de l'objet table view (page 403) affichant la liste des tâches. Les informations incluent la liste des tâches, le nom de la colonne courante et l'ordre de tri de cette colonne. Le gestionnaire retournera ces informations dans un enregistrement, lesquelles sont tout ce dont l'application a besoin pour recréer l'état de la fenêtre courante.

```
on data representation theObject of type ofType
  -- Set up local variables
  set theWindow to window 1 of theObject
  set theDataSource to data source of table view "tasks"
    of scroll view "tasks" of theWindow
  set theTasks to contents of every data cell of every data row
    of theDataSource
  set theSortColumn to sort column of theDataSource
  -- Create a record containing the list of tasks (just a list of lists),
  -- the name of the sort column, and the sort order.
  set theDataRecord to {tasks:theTasks,
```

```
  sortColumnName:name of theSortColumn,
  sortColumnOrder:sort order of theSortColumn}
  return theDataRecord
end data representation
```

Version

Le gestionnaire Data Representation fut ajouté dans la version 1.2 d'AppleScript Studio.

L'application "Task List" fut ajoutée dans la version 1.2 d'AppleScript Studio.

load data representation

Appelé pour charger les données du document lorsque le document est ouvert.

Ce gestionnaire est appelé comme le résultat de l'ouverture par l'utilisateur du panel d'ouverture et de la sélection d'un ou de plusieurs fichiers à ouvrir, ou du glisser-déposer d'un document sur l'icone de l'application ou du double-cliquement de l'icone du document. Les données fournies par ce gestionnaire sont les mêmes que celles que l'application a fourni dans le gestionnaire data representation (page 465) lorsque le document fut enregistré.

Il s'agit d'un gestionnaire de haut niveau que vous pouvez utiliser pour les documents qui sont spécifiques à votre application. Ce gestionnaire chargera alors les données fournies sous la forme que vous aurez choisie (la même forme que celle qui a été précédemment fournie dans le gestionnaire data representation (page 465)). L'application ne devra pas traiter avec l'ouverture du fichier et la lecture des données — AppleScript Studio fournira automatiquement les données au document.

Le contraire de Load Data Representation est data representation (page 465).

Syntaxe

`load data representation`	*reference*	obligatoire
`of type`	*Unicode text*	obligatoire
`with data`	*item*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire Load Data Representation est appelé

`of type` *Unicode text*
: Le type (extension) du fichier document

`with data` *item* (page 60)
: Les données à charger depuis le document

Exemples

L'application "Task List" distribuée depuis la version 1.2 d'AppleScript Studio, fournit le gestionnaire suivant pour montrer le mécanisme de haut niveau permettant la lecture des données des fichiers. Le paramètre `theData` référence l'objet de même type que celui qui fut enregistré par le gestionnaire data representation (page 465) — c'est à dire, un enregistrement contenant la liste des tâches, le nom de la colonne courante et l'ordre de tri de cette colonne. Ce gestionnaire extrait ces informations et les insère dans l'objet data source (page 386) de l'objet table view (page 403) affichant la liste des tâches.

```
on load data representation theObject of type ofType with data theData
  -- Set up local variables
  set theWindow to window 1 of theObject
  set theDataSource to data source of table view "tasks"
    of scroll view "tasks" of theWindow
  -- Restore the sort column and sort order of the data source
  -- based on the information saved
  set sort column of theDataSource
    to data column (sortColumnName of theData) of theDataSource
  set sort order of sort column of theDataSource
    to (sortColumnOrder of theData)
  -- Use the "append" command to quickly populate the data source
  -- with the list of tasks
  append the theDataSource with (tasks of theData)
  -- Return true, signaling success. If you return "false",
  -- the document will fail to load and an alert will be presented.
  return true
end load data representation
```

Version

Le gestionnaire Load Data Representation fut ajouté dans la version 1.2 d'AppleScript Studio.

L'application "Task List" fut ajoutée dans la version 1.2 d'AppleScript Studio.

`read from file`

Appelé lorsque l'application a besoin de lire les données d'un document.

Ce gestionnaire est appelé comme le résultat de l'ouverture par l'utilisateur du panel d'ouverture et de la sélection d'un ou de plusieurs fichiers à ouvrir, ou du glisser-déposer d'un document sur l'icone de l'application ou du double-cliquement de l'icone du document. Les données fournies par ce gestionnaire sont les mêmes que celles qui ont été écrites par le gestionnaire write to file (page 470) lorsque le document fut enregistré.

Il s'agit d'un gestionnaire de bas niveau que vous utiliserez pour travailler avec les documents plus compliqués ou les documents que d'autres applications peuvent lire, comme des fichiers texte. Ce gestionnaire est responsable de l'ouverture du fichier spécifié par le paramètre `path name` transmis et de sa fermeture une fois la lecture finie. Le gestionnaire lit les données fournies en fonction du type fourni par le paramètre `of type`. Voir le gestionnaire write to file (page 470) pour plus d'informations sur les types de document.

Le contraire de Read From File est write to file (page 470).

Syntaxe

`read from file`	*reference*	obligatoire
`of type`	*Unicode text*	obligatoire
`path name`	*Unicode text*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Read From File est appelé

`of type` *Unicode text*
Le type (extension) du fichier

`path name` *Unicode text*
Le chemin (au format POSIX, délimité par des slashs(/)) du fichier à

lire

Exemples

L'application "Plain Text" distribuée depuis la version 1.2 d'AppleScript Studio, montre comment lire et écrire des fichiers texte avec les gestionnaires de bas niveau read from file (page 469) et write to file (page 470), y compris l'ouverture et la fermeture du fichier document.

Version

Le gestionnaire Read From File fut ajouté dans la version 1.2 d'AppleScript Studio.

`write to file`

Appelé lorsque l'application a besoin d'écrire des données dans un document.

Ce gestionnaire est appelé comme le résultat de l'ouverture par l'utilisateur des panels d'enregistrement ("Enregistrer" et "Enregistrer sous...") (ou en utilisant les raccourcis-claviers équivalents) et du choix d'enregistrer le document.

Il s'agit d'un gestionnaire de bas niveau que vous utiliserez pour travailler avec les documents plus compliqués ou les documents que d'autres applications peuvent lire, comme des fichiers texte. Ce gestionnaire est responsable de l'ouverture du fichier spécifié par le paramètre `path name` transmis et de sa fermeture après écriture. Le gestionnaire écrit les données du document en fonction du type fourni par le paramètre `of type`.

Par défaut, le type de document d'une application AppleScript Studio basée sur le modèle "AppleScript Document-based Application" est réglé sur "DocumentType". Vous pouvez changer le type de document de la cible active dans le projet Project Builder en modifiant la section "Document Types" du "Target Editor". Par exemple, dans certaines versions de Project Builder, "Document Types" est situé dans la section "Simple View" de la section "Info.plist Entries". Vous pouvez aussi y effectuer d'autres modifications, comme spécifier l'Extension et les types d'OS. Si vous spécifiez plusieurs types de document, l'application fournira un menu déroulant dans le panneau d'enregistrement (lors de l'enregistrement du document), lequel autorisera l'utilisateur à spécifier le type de document à enregistrer sous.

La valeur choisie dans ce menu est la valeur transmise par le paramètre `of type`.

Le contraire de Write To File est read from file (page 469).

Syntaxe

`write to file`	*reference*	obligatoire
`of type`	*Unicode text*	obligatoire
`path name`	*Unicode text*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Write To File est appelé

`of type` *Unicode text*
Le type (extension) du fichier

`path name` *Unicode text*
Le chemin (au format POSIX, délimité par des slashs(/)) du fichier à lire

Exemples

L'application "Plain Text" distribuée depuis la version 1.2 d'AppleScript Studio, montre comment lire et écrire des fichiers texte avec les gestionnaires de bas niveau read from file (page 469) et write to file (page 470).

Version

Le gestionnaire Write To File fut ajouté dans la version 1.2 d'AppleScript Studio.

Septième partie

Drag and Drop Suite

Cette partie décrit la terminologie de la suite Drag and Drop d'AppleScript Studio, laquelle est disponible depuis la version 1.2.

La suite Drag and Drop définit les termes pour travailler avec le glisser-déposer, comprenant la classe drag info (page 477) pour fournir les informations sur le glisser, et des Events pour glisser, suivre la trace, préparer et conclure le déposer. Pour des informations de même nature, voir la classe pasteboard (page 64).

Le support initial du glisser-déposer fut ajouté dans la version 1.2 d'AppleScript Studio. Il fournit la capacité à divers éléments d'interface de recevoir des Events de glisser. Il n'autorise pas, toutefois, l'engagement d'opérations de glisser-déposer (autres que celles déjà supportées par les classes Cocoa).

Entre autres, les classes supportant le glisser-déposer sont button (page 254), clip view (page 202), color well (page 271), combo box (page 274), control (page 280), image view (page 285), matrix (page 289), movie view (page 296), popup button (page 301), progress indicator (page 306), scroll view (page 213), slider (page 315), stepper (page 320), tab view (page 221), text field (page 325), text view (page 569), et view (page 229).

Pour un objet répondant à n'importe quel gestionnaire de glisser-déposer (les gestionnaires sont décrits en détails dans la section "Events"), vous devrez répertorier le type de glisser que l'objet peut accepter. Vous ferez cela avec la commande register (page 116), en utilisant son paramètre `drag type` pour fournir la liste des types de "pasteboard drag" supportés. Les types de pasteboard possibles sont listés dans la classe pasteboard (page 64).

Le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio. Toutefois, voir

le gestionnaire conclude drop (page 481) pour plus d'informations sur la fourniture du glisser-déposer aux objets text view (page 569) et text field (page 325).

Les classes et Events de la suite Drag and Drop sont décrits dans les chapitres suivants :

Classes 477
Events 481

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Drag and Drop contient la classe suivante :

drag info 477

drag info

Pluriel : `drag infos`
Hérite de : `item` (page 60)
Classe Cocoa : ASKDragInfo

Représente les informations et les données de l'opération de glisser courante.

Un objet drag info est transmis à chaque gestionnaire de glisser-déposer décrit dans la section "Events".

La propriété la plus utile de la classe Drag Info est la propriété *pasteboard*. Elle contient les données pour le glissé, desquelles votre application peut extraire des infos et les utiliser avec un gestionnaire drop (page 486) ou autre.

Propriétés des objets de la classe Drag info

En plus des propriétés qu'il hérite de la classe item (page 60), un objet drag info possède ces propriétés :

destination window

Accès : lecture uniquement

Classe : *window* (page 76)
La fenêtre destinataire de l'opération de glisser

image

Accès : lecture uniquement
Classe : *n'importe*
L'image glissée

image location

Accès : lecture uniquement
Classe : *point*
Non supportée dans la version 1.3 d'AppleScript Studio ; l'emplacement de l'image glissée ; l'emplacement est exprimé sous forme d'une liste de deux nombres {gauche, bas} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

location

Accès : lecture uniquement
Classe : *point*
Non supportée dans la version 1.3 d'AppleScript Studio ; l'emplacement courant dans la fenêtre destinataire de l'opération de glisser, exprimé sous forme d'une liste de deux nombres {gauche, bas} ; chaque fenêtre a son propre système de coordonnées, avec l'origine dans le coin inférieur gauche ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

pasteboard

Accès : lecture uniquement
Classe : *pasteboard* (page 64)
Le pasteboard contenant les données glissées

sequence number

Accès : lecture uniquement
Classe : *integer*
Non supportée dans la version 1.3 d'AppleScript Studio ; l'identificateur unique de l'opération de glisser

source

Accès : lecture uniquement
Classe : *item* (page 60)

Le source des données glissées

source mask

Accès : lecture uniquement
Classe : *integer*
Non supportée dans la version 1.3 d'AppleScript Studio ; définit le type de glissé ("drag operation link", "drag operation copy", "drag operation generic")

Events supportés par les objets de la classe Drag Info

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

L'exemple suivant montre une des manières d'examiner un objet drag info (page 477) dans un gestionnaire drop (page 486). Ce gestionnaire retournera `false`, annulant le déposer, si l'objet pasteboard (page 64) de cet objet ne correspond pas aux types autorisés, ici la vérification porte sur le type "string". Si ce type est présent, le gestionnaire utilise les données pour régler le titre de l'objet dont les données sont déposées et retournera `true` pour finir l'opération de glisser.

```
on drop theObject drag info dragInfo
  set dropped to false
  if "string" is in types of pasteboard of dragInfo then
    set title of theObject to contents of pasteboard of dragInfo
    set dropped to true
  end if
  return dropped
end drop
```

Version

La classe Drag Info fut ajoutée dans la version 1.2 d'AppleScript Studio.

Les propriétés *image location*, *location*, *sequence number* et *source mask* de cette classe ne sont pas supportées dans la version 1.3 d'AppleScript Studio.

Chapitre 2

Events

Les objets basés sur les classes de la suite Drag and Drop supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

conclude drop 481
drag 483
drag entered 483
drag exited 484
drag updated 485
drop 486
prepare drop 488

`conclude drop`

Appelé pour finir une opération de déposer réussi.

Ce gestionnaire est uniquement appelé si les gestionnaires prepare drop (page 488) et drop (page 486) ont tous les deux réussi. Vous pouvez utiliser ce gestionnaire, par exemple, pour nettoyer tout état qui a été réglé ou modifié dans le gestionnaire Prepare Drop.

Le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio.

Les classes text view (page 569) et text field (page 325) ont toutes les

deux intégré le support du glisser-déposer, fourni automatiquement par les classes Cocoa sur lesquelles elles sont basées (NSTextView et NSTextField). Si votre application a uniquement besoin du support du déposer de texte, vous n'aurez besoin de rien de plus. Toutefois, une application voulant gérer les données déposées sur un objet text view ou text field, et n'autorisant pas le support par défaut à les gérer, devra suivre ces étapes :

- Répertoriez les "drag types" que les text view (page 569) ou text field (page 325) peuvent gérer ; voir la commande register (page 116) pour plus de détails
- Connectez le gestionnaire drop (page 486) afin de gérer le texte déposé du text view ou du text field
- Connectez un gestionnaire vierge Conclude Drop pour l'objet text view ou le text field ; cela empêchera les objets texte de fournir leur propre support pour le texte déposé

Syntaxe

`conclude drop`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire Conclude Drop est appelé

`drag info` *drag info* (page 477)
: Les informations à propos de l'opération de glisser

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Conclude Drop à un objet, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge. Votre gestionnaire pourra prendre n'importe quelle décision nécessaire afin de s'occuper de la conclusion du déposer non prise en charge par le gestionnaire drop (page 486). Par exemple, le gestionnaire pourrait nettoyer tout état qui a été réglé ou modifié dans le gestionnaire prepare drop (page 488).

```
on conclude drop theObject drag info dragInfo
  (* Statements to deal with the concluded drop. *)
end conclude drop
```

Voir aussi la description de cette classe ci-dessus.

Version

Le gestionnaire Conclude Drop fut ajouté dans la version 1.2 d'AppleScript Studio.

drag

Non supporté dans la version 1.3 d'AppleScript Studio.

Le support de ce gestionnaire est planifié pour une future version d'AppleScript Studio.

Syntaxe

`drag`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Drop est appelé

`drag info` *drag info* (page 477)
Les informations à propos de l'opération de glisser

Version

Le gestionnaire Drag fut ajouté dans la version 1.2 d'AppleScript Studio, bien qu'il ne fasse rien dans cette version.

drag entered

Appelé lorsqu'un utilisateur glisse le type répertorié de données dans les limites de l'objet.

Votre application pourra ne pas avoir besoin du gestionnaire Drag Entered, étant donné que le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio.

Syntaxe

`drag entered`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Drag Entered est appelé

`drag info` *drag info* (page 477)
Les informations à propos de l'opération de glisser

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Drag Entered à un objet, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge. Le paramètre `theObject` référence l'objet pour lequel un glisser potentiel est entré dans ses limites. Le paramètre `dragInfo` fournit l'accès à l'objet drag info (page 477) contenant toutes les informations pertinentes sur l'opération de glisser.

```
on drag entered theObject drag info dragInfo
  (* Statements to deal with the drag entering. *)
end drag entered
```

Version

Le gestionnaire Drag Entered fut ajouté dans la version 1.2 d'AppleScript Studio.

drag exited

Appelé lorsqu'un utilisateur glisse le type répertorié de données hors des limites de l'objet.

Votre application pourra ne pas avoir besoin du gestionnaire Drag Exited, étant donné que le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio.

Syntaxe

`drag exited`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Drag Exited est appelé

`drag info` *drag info* (page 477)
Les informations à propos de l'opération de glisser

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Drag Exited à un objet, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge. Le paramètre `theObject` référence l'objet pour lequel un glisser potentiel est sorti de ses limites. Le paramètre `dragInfo` fournit l'accès à l'objet drag info (page 477) contenant toutes les informations pertinentes sur l'opération de glisser.

```
on drag exited theObject drag info dragInfo
  (* Statements to deal with the drag exiting. *)
end drag exited
```

Version

Le gestionnaire Drag Exited fut ajouté dans la version 1.2 d'AppleScript Studio.

drag updated

Appelé lorsqu'un utilisateur a glissé un type répertorié de données à l'intérieur des limites de l'objet.

Votre application pourra ne pas avoir besoin du gestionnaire Drag Updated, étant donné que le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio.

Syntaxe

`drag updated`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Drag Updated est appelé

`drag info` *drag info* (page 477)
Les informations à propos de l'opération de glisser

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Drag Updated à un objet, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge. Le paramètre `theObject` référence l'objet dans les limites duquel un glisser avec un type répertorié a bougé. Le paramètre `dragInfo` fournit l'accès à l'objet drag info (page 477) contenant toutes les informations pertinentes sur l'opération de glisser. Notez toutefois, que dans la version 1.3 d'AppleScript Studio, vous ne pouvez pas accéder à la propriété *location* d'un objet drag info (page 477), aussi la capacité à déterminer un point d'insertion est limitée.

```
on drag updated theObject drag info dragInfo
  (* Statements to deal with the drag updating. *)
end drag updated
```

Version

Le gestionnaire Drag Updated fut ajouté dans la version 1.2 d'AppleScript Studio.

drop

Appelé lorsqu'un utilisateur a déposé des données avec le type répertorié sur l'objet.

Vous retournerez `false` depuis ce gestionnaire pour annuler l'opération ; sinon, vous devrez retourner `true` pour signaler le succès de l'opération.

Vous pouvez examiner la propriété *pasteboard* du paramètre `drag info` pour obtenir les données du déposer dans le format requis. Si vous utilisez le gestionnaire prepare drop (page 488) pour vérifier que les données dont vous avez besoin sont présentes, vous ne devrez pas oublier que les données ne seront disponibles que lorsque le gestionnaire Drop sera appelé.

Le gestionnaire Drop est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio. Toutefois, voir le gestionnaire conclude drop (page 481) pour des informations sur la fourniture du glisser-déposer aux objets text view (page 569) et text field (page 325).

Syntaxe

`drop`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
: La référence de l'objet dont le gestionnaire Drop est appelé pour recevoir les données déposées

`drag info` *drag info* (page 477)
: Les informations à propos de l'opération de déposer

Exemples

L'application "Drag and Drop" distribuée depuis la version 1.2 d'AppleScript Studio, montre comment divers objets peuvent accepter le déposer de données. Le gestionnaire Drop suivant, extrait du fichier `Text.applescript` de cette application, montre comment accepter le texte déposé dans un objet text field (page 325).

Le gestionnaire Drop est appelé lorsque le type approprié de données est déposé sur l'objet. Toutes les informations pertinentes sur le déposer sont contenues dans l'objet drag info (page 477) (transmis dans le paramètre `drag info`). Dans ce cas, le gestionnaire vérifie juste que l'objet pasteboard (page 64) de l'objet drag info correspond aux types autorisés, ici la vérification porte sur le type "string". Si ce type est présent, le gestionnaire l'utilise pour régler le contenu de l'objet `text field` transmis (par le paramètre `theObject`).

```
on drop theObject drag info dragInfo
  -- We are only interested in the "string" data type
    -- If that type is present, set the contents of the text
    -- field to the contents of the pasteboard
  if "string" is in types of pasteboard of dragInfo then
    set string value of theObject to contents of pasteboard of dragInfo
  end if
```

```
  return true
end drop
```

Par défaut, la propriété *preferred type* d'un pasteboard est "string", voila pourquoi le gestionnaire ci-dessus ne règle pas le type préféré. Pour explicitement régler le type préféré d'un pasteboard (dans cet exemple, le pasteboard "general") sur "string" avant d'obtenir les données du pasteboard, vous pouvez utiliser les instructions suivantes :

```
set preferred type of pasteboard "general" to "string"
if "string" is in types of pasteboard "general" then
  set myString to contents of pasteboard "general"
```

Version

Le gestionnaire Drop fut ajouté dans la version 1.2 d'AppleScript Studio.

prepare drop

Appelé lorsqu'un utilisateur a déposé les données glissées sur l'objet.

Vous retournerez `false` depuis ce gestionnaire pour annuler l'opération de déposer ; sinon, vous devrez retourner `true` pour continuer l'opération de déposer.

Vous pouvez examiner la propriété *pasteboard* du paramètre `drag info` pour, par exemple, déterminer si elle contient des données au format voulu.

Votre application pourra ne pas avoir besoin du gestionnaire Prepare Drop, étant donné que le gestionnaire drop (page 486) est le seul gestionnaire requis par le support du déposer de données dans AppleScript Studio.

Syntaxe

`prepare drop`	*reference*	obligatoire
`drag info`	*drag info*	obligatoire

Paramètres

reference
La référence de l'objet dont le gestionnaire Prepare Drop est appelé

`drag info` *drag info* (page 477)
Les informations à propos de l'opération de déposer

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Prepare Drop à un objet, AppleScript Studio ajoute automatiquement au script désigné un gestionnaire vierge. Le paramètre `theObject` référence l'objet pour lequel un déposer est sur le point de se faire. Le paramètre `dragInfo` fournit l'accès à l'objet drag info (page 477) contenant toutes les informations pertinentes sur l'opération de glisser. Votre gestionnaire pourra faire n'importe quelle préparation nécessaire, comme être sûr que les données nécessaires pour le déposer sont bien présentes. Le gestionnaire devra retourner `false` pour annuler l'opération de déposer ou `true` pour l'autoriser.

```
on prepare drop theObject drag info dragInfo
  (* Statements to prepare for a drop.
    In this example, only allow drop if string info available. *)
  if "string" is in types of pasteboard of dragInfo then
    return true
  else
    return false
  end if
end prepare drop
```

Version

Le gestionnaire Prepare Drop fut ajouté dans la version 1.2 d'AppleScript Studio.

Huitième partie

Menu Suite

Cette partie décrit la terminologie de la suite Menu d'AppleScript Studio.

La suite Menu définit un petit nombre de classes et de gestionnaires pour le travail avec les menus. Cette suite décrit les classes menu (page 495) et menu item (page 499) et les gestionnaires choose menu item (page 505) et update menu item (page 506) dans les chapitres suivants :

Classes 495
Events 505

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Menu contient les classes suivantes :

menu 495
menu item 499

`menu`

Pluriel : `menus`
Hérite de : `item` (page 60)
Classe Cocoa : NSMenu

Représente un menu.

Pour chaque menu de la barre de menus (comme “Application”, “Fichier”, “Édition”, etc), il y a un objet menu. Pour chaque élément de menu d’un menu (comme “Nouveau” ou “Ouvrir...”), il y a un objet menu item (page 499).

Les illustrations 8.1 et 8.2 montrent également les menus par défaut d’une application AppleScript Studio (lorsque vous créez le projet en choisissant soit le modèle “AppleScript”, soit le modèle “AppleScript Document-based Application” dans Xcode). Les menus sont montrés dans une fenêtre Nib d’Interface Builder, avec le menu “File” ouvert.

Vous pouvez ajouter un menu dans Interface Builder en glissant un élément “Submenu” du panneau “Cocoa-Menus” sur le menu principal. Vous pouvez régler les divers attributs des menus dans la fenêtre Info d’Interface Builder.

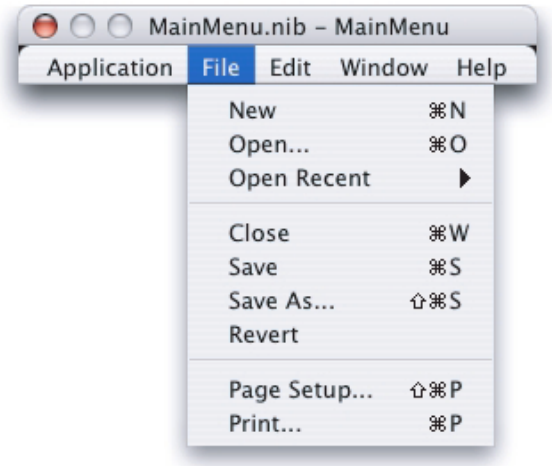

FIG. 8.1 - *Le menu "File" de la fenêtre Nib d'Interface Builder*

Il est possible de modifier dynamiquement un menu en ajoutant ou en supprimant des éléments. Vous pouvez aussi dynamiquement peupler le menu d'un menu déroulant, comme dans la section "Exemples" de la classe popup button (page 301).

Depuis la version 1.3 d'AppleScript Studio, vous pouvez créer des menus et des éléments de menu en utilisant `make new`, et vous pouvez assigner un script aux éléments créés. Pour un exemple de script, voir la section "Exemples" de cette classe.

Pour plus d'informations sur les menus, voir Application Menus et Pop-up Lists dans la documentation Cocoa.

Propriétés des objets de la classe Menu

Un objet menu possède ces propriétés :

auto enables items

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le menu auto-active ses éléments ? Par défaut, cette propriété vaut `true` ; vous pouvez régler cette valeur dans Interface Builder ; toutefois, votre application n'aura pas de solution pratique pour activer et désactiver les éléments de menu des menus standards (comme les menus Fichier et Édition), aussi régler cette propriété sur `false` pour ces menus n'est pas recommandé ; pour en savoir plus sur

le support sous-jacent de l'activation des menus par Cocoa, voir NSMenu et NSMenu Validation.

super menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu contenant ce menu

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre du menu ; vous pouvez régler cette valeur dans Interface Builder

Éléments des objets de la classe Menu

Un objet menu peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`menu` (page 495)
spécifier par : "Les formes-clés standards" (page 15)
Les sous-menus du menu

`menu item` (page 499)
spécifier par : "Les formes-clés standards" (page 15)
Les éléments de menu du menu

Events supportés par les objets de la classe Menu

Un objet menu supporte les gestionnaires répondant aux Events suivants :

Nib

`awake from nib` (page 123)

Exemples

Les instructions suivantes, prévues pour tourner dans l'application Éditeur de Scripts, visent les menus de l'application "TestApp", application totalement fictive. Vous pouvez utiliser ces mêmes instructions dans le

script de votre application AppleScript Studio, bien que vous n'aurez pas besoin de les insérer dans un bloc `tell application`.

```
tell application "TestApp"
  first menu -- result: main menu of application "TestApp"
  title of main menu -- result: "MainMenu"
  menu items of main menu -- result: a list of menu items
  title of menu items of main menu
  -- result: {"", "File", "Edit", "Window", "Help"}
  menus of main menu -- result: long list
  -- {sub menu of menu item id 1 of main menu of application
    "TestApp", etc. }
  menu items of sub menu of menu item id 1 of main menu
  -- result: a long list
end tell
```

L'exemple suivant montre comment créer un script objet et un nouveau menu, y ajouter un élément de menu et assigner le script au nouvel élément.

```
-- Create a script object
  script displayScript
    on choose menu item theItem
      display dialog "Testing"
    end choose menu item
  end script

-- Make a new menu
  set testMenu to make new menu at end of menus of main menu
    with properties  {title:"Testing"}

-- Make a new menu item in the menu
  set displayMenuItem to make new menu item at end of menu items
    of testMenu with properties {title:"Display Test", name:"display"}

-- Assign the script to the menu item
  set script of displayMenuItem to displayScript
```

La section "Exemples" de la classe menu item (page 499) montre comment accéder à des propriétés supplémentaires.

menu item

Pluriel : `menu items`
Hérite de : `item` (page 60)
Classe Cocoa : NSMenuItem

Représente un élément de menu.

Chaque élément de menu est associé à un objet menu (page 495). L'illustration 8.2 montre l'élément de menu "New" du menu "File" de la fenêtre Nib d'Interface Builder.

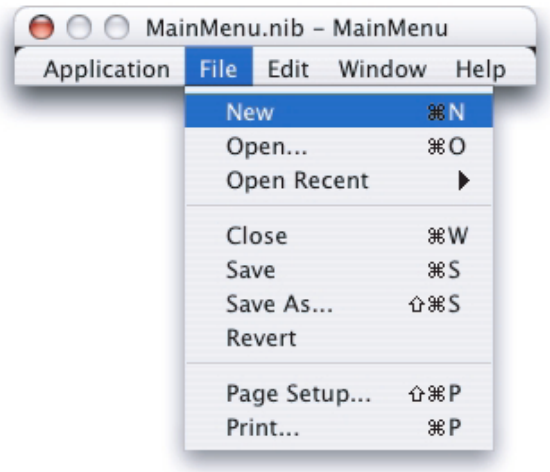

FIG. 8.2 - *L'élément de menu "New" du menu "File" de la fenêtre Nib d'Interface Builder*

Vous pouvez ajouter un élément de menu à un menu dans Interface Builder, en glissant un objet "Item" du panneau "Cocoa-Menus" sur le menu concerné. Vous pouvez régler les divers attributs des éléments de menu dans la fenêtre Info d'Interface Builder. Depuis la version 1.3 d'AppleScript Studio, vous pouvez aussi utiliser `make new` pour créer de nouveaux éléments de menu dans un script, comme l'indique la description de la classe menu (page 495).

Pour plus d'informations sur les menus, voir Application Menus et Pop-up Lists dans la documentation Cocoa.

Propriétés des objets de la classe Menu Item

Un objet menu item possède ces propriétés :

associated object

Accès : lecture / écriture
Classe : *item* (page 60)
L'objet associé avec l'élément de menu

enabled

Accès : lecture / écriture
Classe : *boolean*
L'élément de menu est-il actif ? Vous pouvez connecter un gestionnaire update menu item (page 506) à cet élément de menu pour obtenir son contrôle, qu'il soit actif ou inactif

has sub menu

Accès : lecture uniquement
Classe : *boolean*
L'élément de menu a-t-il un sous-menu ?

image

Accès : lecture / écriture
Classe : *image* (page 58)
L'image de l'élément de menu ; par défaut, aucune image n'est assignée à un menu

key equivalent

Accès : lecture / écriture
Classe : *Unicode text*
Le raccourci-clavier pour sélectionner l'élément de menu ; par défaut, aucun raccourci n'est spécifié pour les éléments de menu ajoutés dans Interface Builder, mais vous pouvez le faire dans le panneau "Attributes" de la fenêtre Info ; les éléments de menu disponibles par défaut ont des raccourci-claviers (comme Cmd + N pour l'élément de menu "Nouveau" du menu "Fichier")

key equivalent modifier

Accès : lecture / écriture
Classe : *number*
Non supportée dans la version 1.3 d'AppleScript Studio ; la touche de fonction du raccourci-clavier ; par défaut, aucun raccourci n'est spécifié pour les éléments de menu ajoutés dans Interface Builder, mais vous pouvez le faire dans le panneau "Attributes" de la fenêtre Info ; les éléments de menu disponibles par défaut ont des touches de fonction

dans leur raccourci (comme Maj. + Cmd + P pour l'élément de menu "Mise en Page" du menu "Fichier")

menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le menu contenant l'élément de menu ; lorsque vous créez et modifiez des menus dans Interface Builder, cette propriété est réglée automatiquement

separator item

Accès : lecture / écriture
Classe : *boolean*
L'élément de menu est-il un élément séparateur ? Vous pouvez ajouter des éléments séparateurs dans Interface Builder en glissant un menu vide du panneau "Cocoa-Menus" (aidez vous des bulles d'aide pour trouver cet élément)

state

Accès : lecture / écriture
Classe : *une des constantes* de cell state value (page 178)
L'état de l'élément de menu

sub menu

Accès : lecture / écriture
Classe : *menu* (page 495)
Le sous-menu de l'élément de menu (s'il y a)

tag

Accès : lecture / écriture
Classe : *integer*
La marque de l'élément de menu (un élément arbitraire associé à l'élément de menu) ; par défaut, cette propriété vaut 0 ; vous pouvez régler cette propriété dans la fenêtre Info d'Interface Builder ; vous pouvez utiliser une marque, par exemple, pour identifier un élément de menu particulier

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre de l'élément de menu ; vous pouvez régler cette propriété dans

la fenêtre Info d'Interface Builder

Commandes supportées par les objets de la classe Menu Item

Votre script peut envoyer la commande suivante à un objet menu item :

`perform action` (page 337)

Events supportés par les objets de la classe Menu Item

Un objet menu item supporte les gestionnaires répondant aux Events suivants :

Menu

`choose menu item` (page 505)

`update menu item` (page 506)

Nib

`awake from nib` (page 123)

Exemples

Les instructions suivantes, prévues pour tourner dans l'application Éditeur de Scripts, montrent comment accéder aux propriétés de l'élément de menu "New" du menu "File" d'une application AppleScript Studio. Vous pouvez utiliser ces instructions dans le script d'une application AppleScript Studio, bien que vous n'aurez pas besoin de les insérer dans un bloc `tell application`.

```
tell application "TestApp"
  set menuItem to second menu item of main menu
  title of menuItem -- result: "File"
  set item1 to first menu item of sub menu of menuItem
  title of item1 -- result: "New"
  key equivalent of item1 -- result: "n"
end tell
```

Pour un exemple montrant comment ajouter un script à un objet menu item, voir la section "Exemples" de la classe menu (page 495).

Version

La propriété *key equivalent modifier* de cette classe n'est pas supportée dans la version 1.3 d'AppleScript Studio.

Chapitre 2

Events

Les objets basés sur les classes de la suite Menu supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

choose menu item 505
update menu item 506

`choose menu item`

Appelé lorsqu'un élément de menu est choisi.

Syntaxe

`choose menu item` *reference* obligatoire

Paramètres

reference

La référence de l'objet menu item dont le gestionnaire Choose Menu Item a été appelé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Choose Menu Item à un objet menu item (page 499), AppleScript Studio ajoute

automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Ce gestionnaire est là où votre application traite avec les choix de menu de l'utilisateur.

```
on choose menu item theObject
  (* Add script statements here to handle the chosen menu item. *)
end choose menu item
```

update menu item

Appelé périodiquement lorsque l'état d'un élément de menu peut avoir besoin d'être mis à jour.

Ce gestionnaire devra retourner `true` pour activer l'élément de menu ou `false` pour le désactiver.

Syntaxe

`update menu item` *reference* obligatoire

Paramètres

reference

La référence de l'objet menu item dont le gestionnaire Update Menu Item a été appelé

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Update Menu Item à un objet menu item (page 499), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le paramètre `theObject` référence l'élément de menu choisi et vous pouvez utiliser ce paramètre pour accéder aux propriétés ou aux éléments de l'élément. Par exemple, vous avez créé une application "Explorateur" permettant de naviguer dans les volumes, si votre application possède un élément de menu pour afficher la date de modification d'un fichier sélectionné, vous pourriez utiliser un gestionnaire Update Menu Item pour désactiver l'élément de menu lorsqu'aucun fichier n'est sélectionné.

```
on update menu item theObject
  (* if the menu item should be enabled... *)
```

```
    return true
  (* other statements *)
  (* if the menu item should be disabled *)
    return false
end update menu item
```

Neuvième partie

Panel Suite

Cette partie décrit la terminologie de la suite Panel d'AppleScript Studio.

La suite Panel définit des classes, des commandes et des Events pour négocier avec les dialogues, les alertes et les panels. La plupart des classes de cette suite héritent de la classe window (page 76), soit directement, soit par l'intermédiaire d'une des classes des panels. Un **panel** est un type spécial d'objet window pouvant être facultativement affiché comme un dialogue ou une fenêtre utilitaire. Pour plus d'informations sur les termes utilisés dans la suite Panel, voir "“Panels” contre “Dialogs” et “Windows”" (page 21).

Les classes, commandes et Events de la suite Panel sont décrits dans les chapitres suivants :

Classes 513
Commandes 535
Events 551

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Panel contient les classes suivantes :

alert reply 513
color-panel 514
dialog reply 518
font-panel 520
open-panel 522
panel 526
save-panel 529

`alert reply`

Pluriel : `alert replies`
Hérite de : `personne`
Classe Cocoa : ASKAlertReply

La réponse de la commande display alert (page 540).

Un objet alert reply est identique dans le concept à la classe `dialog reply` définie dans le complément de pilotage (osax) “Standard Additions” d’AppleScript (fichier `StandardAdditions.osax` dans `/System/Library/ScriptingAdditions`).

Voir aussi display alert (page 540) et alert ended (page 551).

Propriétés des objets de la classe Alert Reply

Un objet alert reply possède une seule propriété. Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y régler sa propriété, vous devrez le faire dans un script.

button returned

Accès : lecture uniquement
Classe : *Unicode text*
Le bouton qui a été cliqué pour arrêter l'alerte ; par exemple, "Annuler" si l'utilisateur a cliqué sur le bouton Annuler

Events supportés par les objets de la classe Alert Reply

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Pour un exemple d'objet alert reply, voir la section "Exemples" du gestionnaire alert ended (page 551). L'application "Display Alert" distribuée avec AppleScript Studio utilise aussi un objet alert reply.

color-panel

Pluriel : color-panels
Hérite de : panel (page 526)
Classe Cocoa : NSColorPanel

Fournit une interface standard pour la sélection des couleurs dans une application.

Un objet color panel utilise un objet color well (page 271) pour sélectionner une couleur précise. Pour utiliser un objet color panel dans les scripts de votre application, vous pouvez accéder à la propriété *color panel* associée avec chaque objet application (page 29). Notez que color-panel (avec un tiret) est le nom de la classe, tandis que *color panel* (propriété de la classe application (page 29)) spécifie un objet de cette classe.

Un objet color panel fournit un certain nombre de modes de sélection de couleurs que vous pouvez régler en utilisant les constantes de color panel

mode (page 179). Toutefois, lorsque vous obtenez ou que vous réglez une propriété de couleur d'un objet AppleScript Studio, la couleur sera exprimée au format RVB, une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur. Par exemple, la couleur verte sera exprimée par {0, 65535, 0}.

L'illustration 9.1 montre le panel "Couleurs". Le slider "Opacity" ne sera pas visible tant que vous n'aurez pas réglé la propriété *shows alpha* du color panel sur `true`. Pour plus d'informations sur les couleurs et les objets color panel, voir Using Color dans la documentation Cocoa.

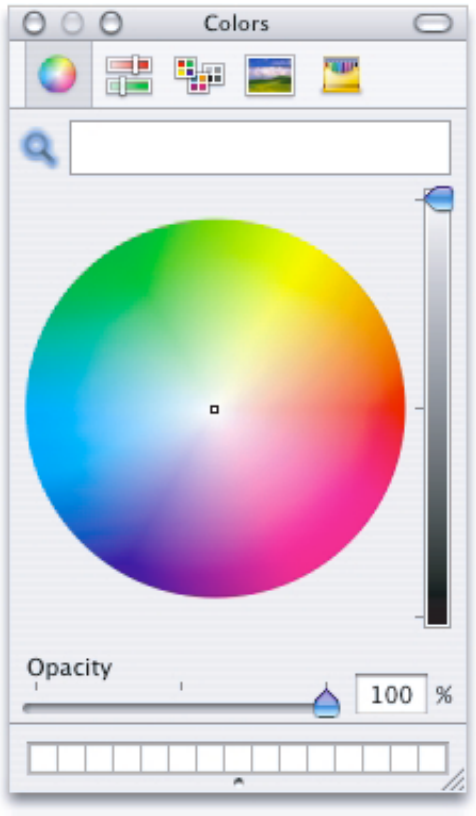

FIG. 9.1 - *Le panel "Couleurs"*

Propriétés des objets de la classe Color-Panel

En plus des propriétés qu'il hérite de la classe panel (page 526), un objet color panel possède ces propriétés. Cette classe n'est pas accessible dans Interface Builder, aussi vous ne pourrez pas y régler ses propriétés, vous devrez le faire dans un script.

alpha

Accès : lecture / écriture
Classe : *real*
La valeur alpha de la couleur ; va de 0.0 (transparent) à 1.0 (opaque) ; par défaut, cette propriété vaut 1.0

color

Accès : lecture / écriture
Classe : *RGB color*
La couleur ; retournée sous la forme d'une liste de trois nombres entiers {valeur rouge, valeur verte, valeur bleue} ; par exemple, {0, 0, 0} représente la couleur noire, tandis que {0, 65535, 0} représente la couleur verte

color mode

Accès : lecture / écriture
Classe : *une des constantes* de color panel mode (page 179)
Le mode couleur de l'objet color panel

continuous

Accès : lecture / écriture
Classe : *boolean*
Faut-il que l'objet color panel renvoie de façon instantanée les modifications de couleur quand l'utilisateur manipule les couleurs dans le panel ? Par défaut, cette propriété vaut `true`

shows alpha

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le panel montre la valeur alpha ? Par défaut, cette propriété vaut `false` ; si vous réglez cette valeur sur `true`, le panel affichera le slider "Opacity" (visible dans l'illustration 9.1) — autrement, il ne l'affichera pas et retournera une valeur alpha égale à 1.0

Éléments des objets de la classe Color-Panel

Un objet color panel peut uniquement contenir les éléments qu'il hérite de la classe panel (page 526).

Events supportés par les objets de la classe Color-Panel

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Un gestionnaire dans une application AppleScript Studio peut rendre visible le panel "Couleurs" de cette application (ou le cacher lorsqu'il est visible) en réglant sa propriété *visible*. Comme les scripts AppleScript Studio visent implicitement l'application, vous n'aurez pas besoin d'une instruction `tell application "MonApplication"` pour accéder à la propriété d'une application.

```
set visible of color panel to true
```

Explication : `color panel` est une des propriétés de la classe application (page 29) et elle a pour classe `color-panel`, `visible` est une propriété de la classe window (page 76) dont hérite la classe panel (page 526) dont hérite à son tour la classe `color-panel`, résultat `visible` est une propriété héritée pour `color panel`. Ce système de poupées russes fait d'AppleScript Studio à la fois un système complexe (untel hérite de untel qui hérite lui d'untel...) et relativement complet (le système d'héritage multipliant les possibilités).

Vous pouvez utiliser l'instruction suivante pour obtenir la couleur sélectionnée dans le panel "Couleurs" :

```
set myColor to color of color panel
-- returns an RGB color value, as a three item list: {int, int, int}
```

Vous pouvez régler la couleur d'un objet color-panel en utilisant cette instruction :

```
set color of color panel to {43591, 0, 2}
```

Le gestionnaire launched (page 135) suivant, connecté à l'objet application (page 29) par l'intermédiaire de l'instance File's Owner dans la fenêtre MainMenu.nib d'Interface Builder, règle la couleur du panel "Couleurs" sur le rouge et rend ce panneau visible lorsque l'application est lancée.

```
on launched theObject
  set color of color panel to {65535, 0, 0}
  set visible of color panel to true
end launched
```

Version

Depuis la version 1.1 d'AppleScript Studio, le nom de la classe `color panel` a été modifié en `color-panel`. Ceci afin de mieux différencier la classe `color-panel` de la propriété *color panel* de l'objet application (page 29).

Avant la version 1.1 d'AppleScript Studio, cette classe avait des fonctions limitées.

dialog reply

Pluriel : `dialog replies`
Hérite de : `personne`
Classe Cocoa : ASKDialogReply

La réponse de la commande display dialog (page 543).

Un objet dialog reply est identique dans le concept à la classe `dialog reply` définie dans le complément de pilotage (osax) "Standard Additions" d'AppleScript (fichier `StandardAdditions.osax` dans `/System/Library/ScriptingAdditions`).

Voir aussi display dialog (page 543) et dialog ended (page 552).

Propriétés des objets de la classe Dialog Reply

Un objet dialog reply possède ces propriétés. Cette classe n'est pas accessible dans Interface Builder, aussi vous ne pourrez pas y régler ses propriétés, vous devrez le faire dans un script.

button returned

Accès : lecture uniquement
Classe : *Unicode text*
Le nom du bouton choisi (vide si le paramètre `giving up after` était renseigné et que le temps imparti pour la réponse écoulé)

gave up

Accès : lecture uniquement
Classe : *boolean*
Le temps imparti pour le dialogue est-il écoulé ? (présent uniquement si le paramètre `giving up after` a été utilisé dans l'appel de la commande display dialog (page 543))

text returned

Accès : lecture uniquement
Classe : *Unicode text*
Le texte saisi (présent uniquement si le paramètre `default answer` était renseigné)

Events supportés par les objets de la classe Dialog Reply

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire dialog ended (page 552) suivant est extrait de l'application "Display Dialog" distribuée avec AppleScript Studio. Cette application montre comment afficher un dialogue et obtenir des informations lorsqu'il est renvoyé. Ce gestionnaire se fait appeler lorsque le dialogue est renvoyé après qu'il fut appelé avec le paramètre optionnel `attached to`, pour qu'il soit montré comme une "feuille". Ce gestionnaire extrait les informations à partir de l'objet dialog reply fourni par le paramètre `dialog reply` et les affiche dans la fenêtre de l'application.

```
on dialog ended theObject with reply theReply
  -- Set the values returned in "theReply"
  set contents of text field "text returned" of window "main" to
    text returned of theReply
  set contents of text field "button returned" of window "main" to
    button returned of theReply
  set state of button "gave up" of window "main" to gave up of theReply
end dialog ended
```

font-panel

Pluriel : font-panels
Hérite de : panel (page 526)
Classe Cocoa : NSFontPanel

Affiche la liste des polices disponibles, autorisant l'aperçu et la sélection de la police.

Il n'y a qu'un seul panel "Polices" par application, accessible par l'intermédiaire de la propriété *font panel* de l'objet application (page 29). Notez que `font-panel` (avec un tiret) est le nom de la classe, tandis que `font panel` spécifie un objet de cette classe.

L'illustration 9.2 montre le panel "Polices". Pour connaître les limitations d'AppleScript Studio dans la gestion des polices, voir la classe font (page 54). Pour d'autres informations sur les polices, voir Font Panels and Font Handling dans la documentation Cocoa.

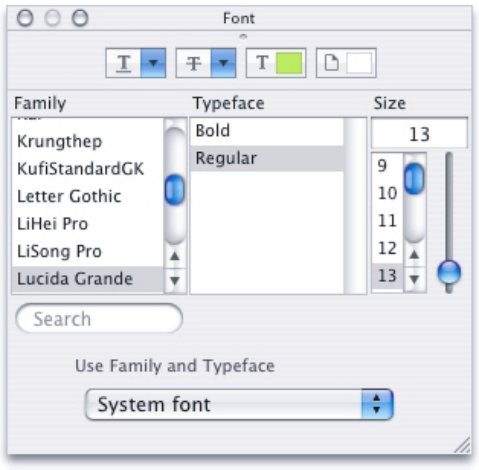

FIG. 9.2 - *Le panel "Polices"*

Propriétés des objets de la classe Font-Panel

En plus des propriétés qu'il hérite de la classe panel (page 526), un objet font panel possède ces propriétés. Cette classe n'est pas accessible dans Interface Builder, aussi vous ne pourrez pas y régler ses propriétés, vous devrez le faire dans un script.

enabled

Accès : lecture / écriture
Classe : *boolean*
Le panel est-il activé ?

font

Accès : lecture / écriture
Classe : *font* (page 54)
La police courante

Éléments des objets de la classe Font-Panel

Un objet font panel peut uniquement contenir les éléments qu'il hérite de la classe panel (page 526).

Events supportés par les objets de la classe Font-Panel

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire d'une application AppleScript Studio peut se référer à la propriété *font panel* de l'objet application (page 29) sans avoir à spécifier l'application, comme dans l'instruction suivante, laquelle affiche le panel "Polices" :

```
set visible of font panel to true
```

Version

Depuis la version 1.1 d'AppleScript Studio, le nom de la classe `font panel` a été modifié en `font-panel`. Ceci afin de mieux différencier la classe `font-panel` de la propriété *font panel* de l'objet application (page 29).

Avant la version 1.1 d'AppleScript Studio, cette classe avait des fonctions limitées.

open-panel

Pluriel : `open-panels`
Hérite de : `save-panel` (page 529)
Classe Cocoa : NSOpenPanel

Fournit un dialogue standard que les applications peuvent utiliser pour demander à l'utilisateur le nom du fichier à ouvrir.

Un objet open panel peut être lancé comme une application modale ou un document modal (comme une "feuille" attachée à une fenêtre). Notez que open-panel (avec un tiret) est le nom de la classe, tandis que *open panel* spécifie un objet de cette classe.

Pour utiliser le panel "Ouvrir" dans une application AppleScript Studio, vous utiliserez la commande display (page 537) pour afficher la propriété *open panel* associée avec chaque objet application (page 29). Si vous affichez ce panel comme une "feuille" (attaché à une fenêtre), vous aurez aussi besoin de connecter un gestionnaire panel ended (page 553). Lorsque l'utilisateur ferme le panel, vous pouvez obtenir des informations, comme la liste des chemins des fichiers que l'utilisateur a sélectionnés, en accédant aux propriétés de l'objet open panel.

Un objet open panel comprend et retourne uniquement des chemins au format POSIX (délimités par des slashs (/)). Il ne comprend pas les types `files` ou `alias`. Vous pouvez, toutefois, utiliser les commandes `POSIX file` et `POSIX path` du complément de pilotage d'AppleScript pour convertir les types de chemins. Ces commandes sont définies dans le complément de pilotage (osax) "Standard Additions" d'AppleScript (fichier `StandardAdditions.osax` dans `/System/Library/ScriptingAdditions`).

Pour plus d'informations, voir save-panel (page 529), ainsi que File Management and Windows and Panels dans la documentation Cocoa.

L'illustration 9.3 montre le panel "Ouvrir".

Propriétés des objets de la classe Open-Panel

En plus des propriétés qu'il hérite de la classe save-panel (page 529), un objet open panel possède ces propriétés (voir la section "Version" de cette classe pour savoir dans quelle version d'AppleScript Studio furent ajoutées certaines propriétés). Cette classe n'est pas accessible dans Interface Builder, aussi vous ne pourrez pas y régler ses propriétés. L'application "Open Panel", distribuée avec AppleScript Studio depuis la version 1.1, montre

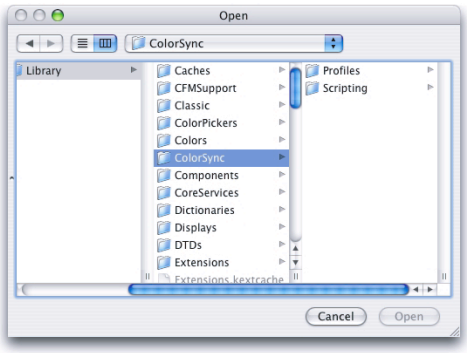

FIG. 9.3 - *Le panel "Ouvrir"*

comment régler la plupart des propriétés du panel "Ouvrir" dans un script.

Propriétés des objets de la classe Open-Panel

allows multiple selection

Accès : lecture / écriture
Classe : *boolean*
Plusieurs éléments peuvent-ils être sélectionnés en même temps ?

can choose directories

Accès : lecture / écriture
Classe : *boolean*
Les répertoires peuvent-ils être sélectionnés ?

can choose files

Accès : lecture / écriture
Classe : *boolean*
Les fichiers peuvent-ils être sélectionnés ?

path names

Accès : lecture uniquement
Classe : *list*
La liste des fichiers sélectionnés devant être ouverts ; chaque nom est un chemin POSIX (délimité par des slashs (/))

Éléments des objets de la classe Open-Panel

Un objet open panel peut uniquement contenir les éléments qu'il hérite de la classe panel (page 526).

Commandes supportées par les objets de la classe Open-Panel

Votre script peut envoyer les commandes suivantes à un objet open-panel :

`display` (page 537)
`display panel` (page 547)

Events supportés par les objets de la classe Open-Panel

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire d'une application AppleScript Studio peut se référer à la propriété *open panel* de l'objet application (page 29) sans avoir à spécifier l'application, comme dans l'instruction suivante, laquelle obtient la liste des chemins POSIX des fichiers choisis dans le panel "Ouvrir" :

```
set fileList to path names of open panel
```

Une application pourra utiliser le gestionnaire clicked (page 348) suivant pour indiquer à l'utilisateur de choisir un dossier. Ce gestionnaire règle d'abord les propriétés du panel "Ouvrir" de l'objet application (page 29) afin de ne spécifier qu'un dossier (ou répertoire). Notez que vous n'avez pas besoin de spécifier implicitement le panel "Ouvrir" comme appartenant à l'objet application.

```
on clicked theObject
  set can choose directories of open panel to true
  set can choose files of open panel to false
  display open panel attached to window "main"
end clicked
```

Comme le gestionnaire affiche le panel comme une "feuille", le panel est un document modal, ce qui signifie que l'exécution de l'application continue après que le panel soit affiché. Aussi pour obtenir le choix de l'utilisateur du panel, l'application a besoin d'utiliser Interface Builder pour connecter un gestionnaire panel ended (page 553) à l'objet window (page 76) afin que le panel soit attaché. Le gestionnaire Panel Ended est appelé lorsque le panel

est renvoyé. Dans l'exemple montré ici, le gestionnaire vérifie le résultat et s'il vaut 1 (un dossier a été choisi ; 0 indique que le panel a été annulé), extrait le dossier de la liste retournée. Comme mentionné au-dessus, le chemin du dossier est un chemin POSIX.

```
on panel ended theObject with result withResult
  if withResult is 1 then
    set theFolder to item 1 of (path names of open panel as list)
    -- do something with the supplied folder path
  end if
end panel ended
```

Si un script n'attache pas le panel à une fenêtre, le panel est affiché comme une application modale, et l'exécution s'arrête jusqu'à ce que le panel soit renvoyé. Dans ce cas, un gestionnaire panel ended (page 553) n'est pas utile.

L'application "Open Panel", distribuée avec AppleScript Studio depuis la version 1.1, montre comment utiliser le panel "Ouvrir", à la fois comme un panel séparé et comme une "feuille". Voir "Document Suite" (page 455) pour des informations sur la manière de lire et d'écrire des fichiers.

Version

La propriété *path names* est apparue avec la version 1.1 d'AppleScript Studio.

Depuis la version 1.1 d'AppleScript Studio, le nom de la classe `open panel` a été modifié en `open-panel`. Ceci afin de mieux différencier la classe `open-panel` de la propriété *open panel* de l'objet application (page 29).

Avant la version 1.1 d'AppleScript Studio, cette classe avait des fonctions limitées.

L'application "Open Panel" a été distribuée pour la première fois avec la version 1.1 d'AppleScript Studio.

panel

Pluriel : panels
Hérite de : window (page 76)
Classe Cocoa : NSPanel

Un type de fenêtres qui a généralement une fonction auxiliaire dans une application.

Par exemple, un panel peut être facultativement affiché comme une fenêtre utilitaire, laquelle peut flotter par dessus d'autres fenêtres. Vous créerez un objet panel dans Interface Builder en le faisant glisser en dehors du panneau "Cocoa-Windows". Les classes color-panel (page 514) et font-panel (page 520) sont basées sur des classes Cocoa qui descendent de NSPanel.

Pour plus d'informations, voir la commande display (page 537), ainsi que Windows and Panels dans la documentation Cocoa.

Propriétés des objets de la classe Panel

En plus des propriétés qu'il hérite de la classe window (page 76), un objet panel possède cette propriété :

floating

Accès : lecture / écriture
Classe : *boolean*
Le panel est-il un panel flottant ?

Éléments des objets de la classe Panel

Un objet panel peut uniquement contenir les éléments qu'il hérite de la classe window (page 76).

Commandes supportées par les objets de la classe Panel

Votre script peut envoyer les commandes suivantes à un objet panel :

close panel (page 535)
display (page 537)
display panel (page 547)

Events supportés par les objets de la classe Panel

Un objet panel supporte les gestionnaires répondant aux Events suivants :

Clavier

`keyboard down` (page 133)
`keyboard up` (page 134)

Souris

`mouse down` (page 137)
`mouse dragged` (page 138)
`mouse entered` (page 139)
`mouse exited` (page 139)
`mouse up` (page 141)
`right mouse down` (page 147)
`right mouse dragged` (page 149)
`right mouse up` (page 150)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

Panel

`alert ended` (page 551)
`dialog ended` (page 552)
`panel ended` (page 553)

Window

`became key` (page 127)
`became main` (page 128)
`exposed` (page 131)
`miniaturized` (page 136)
`moved` (page 142)
`opened` (page 143)
`resigned key` (page 145)
`resigned main` (page 146)
`resized` (page 147)

`should close` (page 151)
`should zoom` (page 156)
`was miniaturized` (page 159)
`will close` (page 161)
`will miniaturize` (page 164)
`will move` (page 164)
`will open` (page 165)
`will resize` (page 167)
`will zoom` (page 169)

Exemples

Le gestionnaire clicked (page 348) suivant est extrait partiellement du gestionnaire Clicked de l'application "Display Panel" distribuée avec AppleScript Studio. Ce gestionnaire se trouve dans le fichier `Window.applescript`. Cette application montre comment afficher un panel et obtenir des informations lorsqu'il est renvoyé. Comme ce gestionnaire utilise le paramètre `attached to` pour spécifier que le gestionnaire devra être affiché attaché à la fenêtre "main", l'application fournit un gestionnaire panel ended (page 553) qui est appelé lorsque le panel est renvoyé. Aucun gestionnaire Panel Ended n'est utile si le panel n'est pas attaché, car alors il est affiché comme un document modal et le contrôle continue dans l'instruction après que le panel soit affiché.

```
on clicked theObject
  -- Some statements not shown
  -- Make sure panel has been loaded from nib into global property
  if not (exists panelWIndow) then
    load nib "SettingsPanel"
    set panelWIndow to window "settings"
  end if

  -- Statements for setting state of panel items not shown
  -- Now display the panel
  display panelWIndow attached to window "main"
  -- Other statements not shown
end clicked
```

L'application "Display Panel" utilise l'instruction suivante pour afficher un panel en tant qu'application modale. L'appel de `display` retourne le

nombre du bouton utilisé pour renvoyer le panel, et le résultat est utilisé ici dans une instruction de test (`if...then...`). Une valeur 0 indique que le panel a été annulé ; 1 indique qu'il a été accepté.

```
if (display panelWIndow) is 1 then
```

Note : L'application "Display Panel" utilise la commande display (page 537), manière recommandée pour afficher un panel, plutôt qu'avec la commande display panel (page 547).

save-panel

Pluriel : `save-panels`
Hérite de : `panel` (page 526)
Classe Cocoa : NSSavePanel

Autorise les utilisateurs à spécifier le répertoire et le nom sous lequel un fichier doit être enregistré.

La classe save-panel supporte la navigation dans un système de fichiers et il accepte des visualisations personnalisées. Pour utiliser le panel d'enregistrement dans les scripts de vos applications AppleScript Studio, vous pouvez accéder à la propriété *save panel* qui est associée avec chaque objet application (page 29). Notez que save-panel (avec un tiret) est le nom de la classe, tandis que *save panel* spécifie un objet de cette classe.

Pour utiliser un panel d'enregistrement dans une application AppleScript Studio, vous utiliserez la commande display (page 537) pour afficher la propriété *save panel* associée avec chaque objet application (page 29). Si vous affichez le panel comme une "feuille" (attaché à une fenêtre), vous aurez aussi besoin de connecter un gestionnaire panel ended (page 553). Lorsque l'utilisateur ferme le panel, vous pouvez obtenir des informations, comme le chemin qu'a choisi l'utilisateur pour l'enregistrement d'un fichier, en accédant aux propriétés de l'objet save-panel.

Un objet save panel comprend et travaille uniquement avec des chemins au format POSIX (délimités par des slashs (/)). Il ne comprend pas les types `files` ou `alias`. Vous pouvez, toutefois, utiliser les commandes `POSIX file` et `POSIX path` du complément de pilotage d'AppleScript pour convertir les types de chemins. Ces commandes sont définies dans le complément de pilotage (osax) "Standard Additions" d'AppleScript (fichier `StandardAdditions.osax` dans `/System/Library/ScriptingAdditions`).

L'illustration 9.4 montre un panel d'enregistrement. Pour plus d'informations sur les panels, voir Windows and Panels dans la documentation Cocoa.

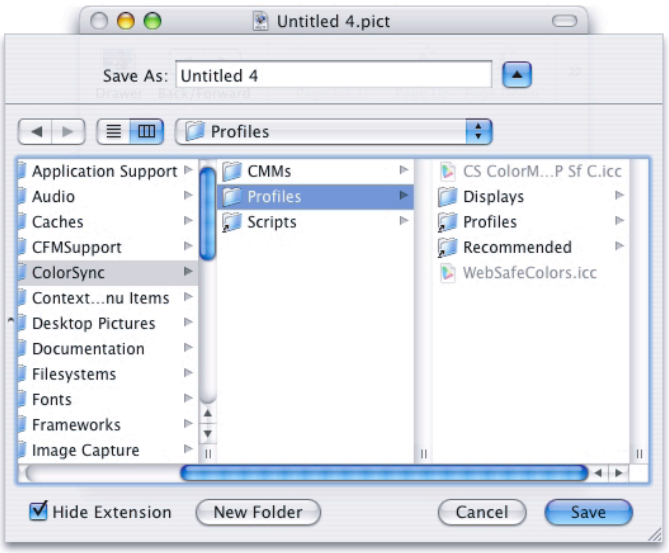

FIG. 9.4 - *Le panel d'enregistrement*

Propriétés des objets de la classe Save-Panel

En plus des propriétés qu'il hérite de la classe panel (page 526), un objet save panel possède ces propriétés. Cette classe n'est pas accessible dans Interface Builder, aussi vous ne pourrez pas y régler ses propriétés. L'application "Save Panel", distribuée avec AppleScript Studio depuis la version 1.1, montre comment régler la plupart des propriétés d'un objet save panel dans un script.

directory

Accès : lecture / écriture
Classe : *Unicode text*
Le répertoire à utiliser dans le panel ; devra être un chemin POSIX (délimité par des slashs (/))

expanded

Accès : lecture uniquement
Classe : *boolean*
Le panel est-il développé ? Si oui, la colonne du navigateur est visible,

comme dans l'illustration 9.4

path name

Accès : lecture uniquement
Classe : *Unicode text*
Le chemin retourné depuis le panel ; un chemin POSIX ; cette propriété s'appelait *file name* avant la version 1.2 d'AppleScript Studio

prompt

Accès : lecture / écriture
Classe : *Unicode text*
Le message à afficher dans le bouton par défaut (par défaut "Save" ou "Enregistrer" pour la version française, comme dans l'illustration 9.4)

required file type

Accès : lecture / écriture
Classe : *Unicode text*
Spécifie l'extension devant être ajoutée aux fichiers sélectionnés n'ayant pas déjà cette extension ; une extension, comme "rtf" ou "txt" ; n'inclut pas le point commençant l'extension

title

Accès : lecture / écriture
Classe : *Unicode text*
Le titre du panel ("Save as" dans l'illustration 9.4)

treat packages as directories

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le panel traite les packages comme des répertoires ? Dans Mac OS X, une application est packagée comme un bundle, ou un répertoire dans le système de fichiers qui stocke le code exécutable et les ressources en relation avec ce code ; "package" est quelque fois utilisé comme un synonyme de "bundle" ; si cette propriété est réglée sur `false`, le panel d'enregistrement montrera le contenu d'un bundle, plutôt que de l'afficher comme s'il s'agissait d'un fichier

Éléments des objets de la classe Save-Panel

Un objet save panel peut uniquement contenir les éléments qu'il hérite de la classe panel (page 526).

Commandes supportées par les objets de la classe Save-Panel

Votre script peut envoyer les commandes suivantes à un objet save panel :

`display` (page 537)
`display panel` (page 547)

Events supportés par les objets de la classe Save-Panel

Cette classe n'est pas accessible dans Interface Builder, par conséquent vous ne pourrez pas y connecter de gestionnaires.

Exemples

Le gestionnaire d'une application AppleScript Studio peut se référer à la propriété *open panel* de l'objet application (page 29) sans avoir à spécifier l'application, comme dans l'instruction suivante, laquelle affiche le panel d'enregistrement (attaché à la fenêtre ayant comme nom AppleScript "main") :

```
display save panel attached to window "main"
```

Comme l'instruction affiche le panel comme une "feuille", le panel sera un document modal, ce qui signifie que l'exécution de l'application continuera après que le panel soit affiché. Aussi pour obtenir le choix de l'utilisateur à partir du panel, l'application aura besoin d'utiliser Interface Builder pour connecter un gestionnaire panel ended (page 553) à l'objet window (page 76) auquel le panel est attaché. Le gestionnaire Panel Ended sera appelé lorsque le panel sera renvoyé.

Si un script n'attache pas le panel à la fenêtre, le panel est affiché en tant qu'application modale, et donc l'exécution s'arrête jusqu'à ce que le panel soit renvoyé. Dans ce cas là, un gestionnaire Panel Ended ne sera pas utile.

Pour un résumé sur la manière d'utiliser le panel d'enregistrement, voir la section "Exemples" de la commande display (page 537) et l'application "Save Panel" distribuée avec AppleScript Studio. Pour un exemple qui montre comment utiliser un panel d'enregistrement, voir la section "Exemples" de la classe open-panel (page 522). Voir également "Document Suite" (page 455) pour des informations sur la manière de lire et d'écrire des fichiers.

Version

Depuis la version 1.1 d'AppleScript Studio, le nom de la classe `save panel` a été modifié en `save-panel`. Ceci afin de mieux différencier la classe `save-panel` de la propriété *save panel* de l'objet application (page 29).

Avant la version 1.1 d'AppleScript Studio, cette classe avait des fonctions limitées.

Avant la version 1.1 d'AppleScript Studio, le nom de la propriété *path name* était *file name*.

L'application "Save Panel" a été distribuée pour la première fois avec la version 1.1 d'AppleScript Studio.

Chapitre 2

Commandes

Les objets basés sur les classes de la suite Panel supportent les commandes suivantes. Une **commande** est un mot ou une phrase qu'un script peut utiliser pour demander une action. Pour déterminer les commandes supportées par chaque classe, voir les descriptions propres à chaque classe.

close panel 535
display 537
display alert 540
display dialog 543
display panel 547
load panel 548

`close panel`

Ferme le panel spécifié, retournant facultativement une valeur.

Syntaxe

`close panel`	*reference*	obligatoire
`[with result]`	*n'importe*	facultatif

Paramètres

reference

La référence du panel à fermer

`[with result]` *n'importe*

Le résultat du panel ; doit être soit un nombre ou une chaîne de caractères pouvant être convertie en nombre ; par exemple, un nombre entier indiquant le bouton appuyé

Exemples

Le gestionnaire clicked (page 348) suivant est extrait de l'application "Display Panel", distribuée avec AppleScript Studio, et il est connecté aux boutons du panel de cette application. Ce gestionnaire se trouve dans le fichier `Settings.applescript`. L'application "Display Panel" montre comment afficher un panel et obtenir des informations lorsqu'il est renvoyé.

Si le panel est affiché comme une "feuille" (document modal), l'exécution de l'application continue après que le panel soit affiché. Pour obtenir le choix de l'utilisateur dans le panel, l'application connecte un gestionnaire panel ended (page 553) à l'objet application (page 29) auquel le panel est attaché (comme le décrit la section "Exemples" de la classe panel (page 526)). Le gestionnaire Panel Ended est appelé lorsque le panel est renvoyé. La valeur retournée par la commande Close Panel est transmise au gestionnaire Panel Ended, lequel ne fait rien tant que la valeur est 1, indiquant que l'utilisateur a cliqué sur le bouton "Change" (et non le bouton "Cancel").

Si le panel avait été affiché comme une fenêtre séparée (application modale), l'exécution s'arrête jusqu'à ce que le panel soit renvoyé. Dans ce cas là, un gestionnaire panel ended (page 553) n'est pas utile. La valeur retournée par la commande Close Panel est directement envoyée à l'appel du script, lequel encore ne fait rien tant que la valeur est 1, indiquant que l'utilisateur a cliqué le bouton "Change" (et non le bouton "Cancel").

```
on clicked theObject
  if name of theObject is "cancel" then
    close panel (window of theObject)
  else if name of theObject is "change" then
    close panel (window of theObject) with result 1
  end if
end clicked
```

display

Présente un panel à la mode application modale, ou document modal (comme une “feuille”) lorsque le paramètre optionnel `attached to` est utilisé.

Vous devrez utiliser Display plutôt que display panel (page 547) pour afficher les objets des classes open-panel (page 522), save-panel (page 529), color-panel (page 514) et font-panel (page 520). Pour des exemples montrant la terminologie correcte, voir la section “Exemples” ci-dessous.

La commande Display autorise votre application à afficher les panels d’ouverture et d’enregistrement qui ont été ajoutés comme propriétés de l’objet application (page 29) dans la version 1.1 d’AppleScript Studio.

Syntaxe

`display`	*reference*	obligatoire
`[afterwards calling]`	*n’importe*	facultatif
`[attached to]`	*window*	facultatif
`[for file types]`	*list*	facultatif
`[in directory]`	*Unicode text*	facultatif
`[with file name]`	*Unicode text*	facultatif

Paramètres

reference
: La référence de la fenêtre à afficher (auquel cas, vous utiliserez une instruction telle que `display panelWindow` ; pour le panel d’enregistrement ou d’ouverture, utilisez `display save panel` ou `display open panel`)

`[afterwards calling]` *n’importe*
: Non supportée dans la version 1.3 d’AppleScript Studio ; la référence du script à lancer lorsque la fenêtre affichée est renvoyée

`[attached to]` *window* (page 76)
: La fenêtre à laquelle attacher la fenêtre affichée ; attacher une fenêtre lui donne le statut de document modal (attachée comme une “feuille” à la fenêtre spécifiée)

`[for file types]` *list*
: La liste d’extensions de fichiers qui sont autorisées (pour les panels d’enregistrement ou d’ouverture), comme “rtf” ou “txt” ; ne pas inclure

le point marquant l'extension

`[in directory]` *Unicode text*
Le répertoire de départ (pour les panels d'enregistrement ou d'ouverture)

`[with file name]` *Unicode text*
Le nom de ficher par défaut (pour les panels d'enregistrement ou d'ouverture)

Résultats

integer
Une valeur indiquant quel bouton a été utilisé pour renvoyer le panel. Pour les panels d'ouverture et d'enregistrement d'AppleScript Studio, 0 représente le bouton "Annuler" et 1 le bouton "Enregistrer" ou "Ouvrir". Si vous utilisez la commande Display avec votre propre panel, vous appellerez la commande close panel (page 535), en transmettant une valeur entière qui est à son tour transmise à votre gestionnaire panel ended (page 553), et indique comment votre panel a été renvoyé. Dans ce cas là, le retour de valeurs est arbitraire, et c'est à votre application de les assigner et de les interpréter

Exemples

Vous suivrez généralement ces étapes, basées sur l'application "Save Panel" distribuée avec AppleScript Studio, pour afficher et répondre à un panel d'ouverture ou d'enregistrement (ici, un panel d'enregistrement) :

- Créer votre application dans Interface Builder.
  - Vous pourriez vouloir afficher le panel lorsque l'utilisateur cliquera sur un bouton dans une fenêtre. Pour faire cela, utilisez un gestionnaire clicked (page 348), comme dans l'application "Save Panel".
  - Ou vous pourriez vouloir afficher le panel lorsque l'utilisateur choisit l'élément "Enregistrer sous..." dans le menu "Fichier". Dans ce cas là, utilisez un gestionnaire choose menu item (page 505).
- Si vous montrez le panel comme une "feuille", connectez un gestionnaire panel ended (page 553) à la fenêtre à laquelle vous attacherez le panel.
- Pour afficher le panel comme une "feuille" (document modal), utilisez le paramètre `attached to` et ne vous attendez pas à obtenir une réponse immédiate. Le gestionnaire Panel Ended sera appelé lorsque

l'utilisateur renverra le panel, et vous pourrez obtenir des informations sur le panel à ce moment là, comme le décrit la section "Exemples" du gestionnaire panel ended (page 553).

Ci-dessous, vous pouvez voir comment l'application "Save Panel" affiche un panel en tant que "feuille" :

```
display save panel in directory theDirectory with file name theFileName
attached to window of theObject
```

Notez que lorsque vous affichez le panel d'enregistrement d'AppleScript Studio, visible dans l'illustration 9.4, vous utilisez la propriété *save panel* de l'objet application (page 29).

- Si vous affichez le panel en suivant la méthode standard (pas en tant que "feuille"), le résultat sera une application modale (rien d'autre ne pourra se produire tant que l'utilisateur n'aura pas répondu) et le contrôle reviendra à l'instruction qui suit l'instruction `display save panel`. Voici comment l'application "Save Panel" fait cela :

  ```
  set theResult to display save panel in directory theDirectory with file
  name theFileName
  ```

  Le résultat est un nombre entier, où 0 représente le bouton "Annuler" et 1 le bouton "Enregistrer".

Vous pouvez aussi utiliser la commande Display pour afficher un panel que vous aurez préalablement construit dans Interface Builder (à la place du panel d'enregistrement standard fourni par AppleScript Studio). Dans ce cas là, vous aurez besoin d'appeler la commande close panel (page 535) lorsque votre panel sera renvoyé. Pour plus de détails sur la manière de faire cela, voir l'application "Display Panel" distribuée avec AppleScript Studio.

Étant donné une fenêtre "main" et un panel "preferences" dans un fichier .nib "preferences.nib", le script suivant affiche le panel attaché à la fenêtre :

```
load nib "preferences"
set preferencesPanel to window "preferences"
display preferencesPanel attached to window "main"
```

Discussion

Lorsque vous affichez une fenêtre en tant qu'application modale, votre script attendra que l'utilisateur renvoie la fenêtre. Si vous affichez la fenêtre en tant que document modal en spécifiant le paramètre `attached to`, l'exécution du script continue. Pour exécuter n'importe quelle action lorsque

l'utilisateur renvoie la fenêtre, vous devrez fournir un gestionnaire panel ended (page 553). Vous pouvez voir un exemple sur ce gestionnaire dans l'application "Display Panel" distribuée avec AppleScript Studio.

Version

La commande Display fut ajoutée dans la version 1.1 d'AppleScript Studio. Avant cette version, vous pouviez utiliser la commande display panel (page 547). Une commande Display différente était disponible dans la suite Application de la version 1.0 d'AppleScript Studio.

Depuis la version 1.2 d'AppleScript Studio, la commande Display est recommandée à la place de la commande display panel (page 547).

Le paramètre `afterwards calling` de cette commande n'est pas supportée dans la version 1.3 d'AppleScript Studio.

L'application "Save Panel" fut distribuée pour la première fois avec la version 1.1 d'AppleScript Studio.

`display alert`

Affiche l'alerte spécifiée.

La plupart des paramètres optionnels permettent de contrôler l'affichage de l'alerte. Vous afficherez généralement une alerte afin de fournir des informations à l'utilisateur et à laquelle il devra répondre immédiatement. Toutefois, vous pouvez afficher une alerte en tant que "feuille" (attachée à la fenêtre), autorisant l'utilisateur à continuer à travailler avec d'autres fenêtres avant de répondre à celle-ci.

L'illustration 9.5 montre un message d'alerte affiché en tant que "feuille" par l'application "Display Alert" distribuée avec AppleScript Studio.

Syntaxe

`display alert`	*Unicode text*	obligatoire
`[afterwards calling]`	*n'importe*	facultatif
`[alternate button]`	*Unicode text*	facultatif
`[as]`	*une constante*	facultatif
`[attached to]`	*window*	facultatif
`[default button]`	*Unicode text*	facultatif
`[message]`	*Unicode text*	facultatif
`[other button]`	*Unicode text*	facultatif

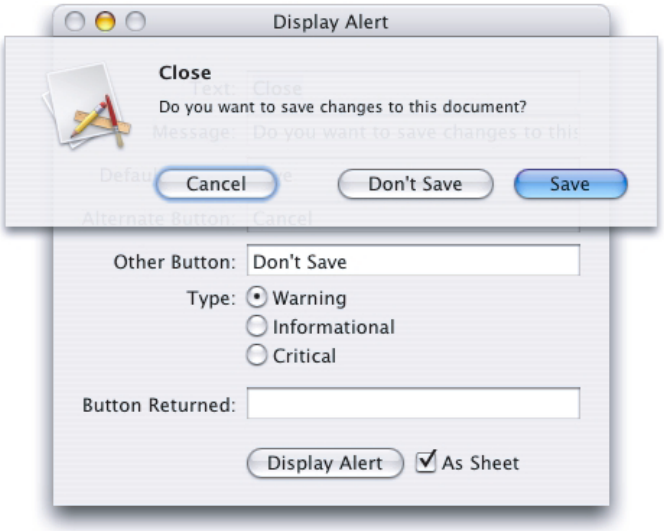

FIG. 9.5 - *Un message d'alerte affiché en tant que "feuille" par la commande Display Alert*

Paramètres

Unicode text
: Le titre du message à afficher dans l'alerte ; peut être une chaîne de caractères vide

`[afterwards calling]` *n'importe*
: Non supportée dans la version 1.3 d'AppleScript Studio ; la référence du script à lancer lorsque l'alerte est finie

`[alternate button]` *Unicode text*
: Le titre du bouton alternatif

`[as]` *une des constantes* de alert type (page 174)
: Le type d'alerte

`[attached to]` *window*
: La fenêtre à laquelle attacher l'alerte

`[default button]` *Unicode text*
: Le titre du bouton par défaut

`[message]` *Unicode text*
: Le texte du message de l'alerte

`[other button]` *Unicode text*
: Le titre de l'autre bouton

Résultats

alert reply

Un objet alert reply (page 513) contenant les informations sur l'alerte renvoyée. Lorsque l'alerte est affichée comme une "feuille" (attachée à la fenêtre), il n'y a pas de résultat immédiat, et vous devrez installer un gestionnaire alert ended (page 551) pour répondre à l'alerte renvoyée.

Exemples

L'instruction suivante montre la syntaxe permettant d'afficher une alerte en tant que "feuille" (laquelle donne comme résultat une alerte en tant que document modal). Elle est extraite du gestionnaire clicked (page 348) de l'application "Display Alert" distribuée avec AppleScript Studio. La plupart des paramètres sont des variables réglées depuis des champs texte dans la fenêtre principale de l'application (en partie visible en arrière-plan dans l'illustration 9.5).

```
display alert dialogText as dialogType message dialogMessage
  default button defaultButtonTitle alternate button alternateButtonTitle
  other button otherButtonTitle attached to window "main"
```

Lorsque vous afficherez une alerte attachée à une fenêtre, vous devrez installer un gestionnaire Alert Ended à la fenêtre. Le gestionnaire sera appelé lorsque l'alerte sera renvoyée. Pour un exemple de gestionnaire Alert Ended, voir la section "Exemples" du gestionnaire alert ended (page 551).

L'instruction suivante montre la syntaxe permettant d'afficher une alerte qui n'est pas attachée à une fenêtre (et sera par conséquent une application modale). De même que plus haut, la plupart des paramètres sont des variables réglées depuis des champs texte dans la fenêtre principale de l'application (en partie visible en arrière-plan dans l'illustration 9.5). Ici, pourtant, l'exécution s'arrête jusqu'à ce que l'alerte soit renvoyée, et Display Alert retourne un objet alert reply (page 513), à partir duquel vous pouvez obtenir des informations sur la manière dont l'alerte a été renvoyée (comme le montre la section "Exemples" du gestionnaire alert ended (page 551)).

```
set theReply to display alert dialogText as dialogType
  message dialogMessage default button defaultButtonTitle
  alternate button alternateButtonTitle other button otherButtonTitle
```

Version

Le paramètre `afterwards calling` de cette commande n'est pas supportée dans la version 1.3 d'AppleScript Studio.

`display dialog`

Affiche le dialogue spécifié.

AppleScript Studio annule la commande `display dialog` du complément de pilotage d'AppleScript (fournie dans le fichier `/System/Library/ScriptingAdditions/StandardAdditions.osax`) pour implémenter sa propre version.

Cette commande a plusieurs paramètres optionnels vous autorisant à contrôler l'affichage du dialogue. Par exemple, vous pouvez présenter la fenêtre en tant qu'application modale ou en tant que document modal (comme une "feuille"). Afficher un dialogue en tant que "feuille" permet à l'utilisateur de continuer à travailler avec d'autres fenêtres avant de répondre.

L'illustration 9.6 montre un dialogue "feuille" affiché par l'application "Display Dialog" distribuée avec AppleScript Studio. Cette application montre comment afficher un dialogue et obtenir des informations lorsqu'il est renvoyé.

Syntaxe

`display dialog`	*string*	obligatoire
`[attached to]`	*window*	facultatif
`[buttons]`	*list*	facultatif
`[default answer]`	*Unicode text*	facultatif
`[default button]`	*n'importe*	facultatif
`[giving up after]`	*integer*	facultatif
`[with icon]`	*n'importe*	facultatif

Paramètres

string

Le texte du dialogue à afficher

`[attached to]` *window* (page 76)

La fenêtre à laquelle est attaché le dialogue ; fournir ce paramètre fait que le dialogue est affiché en tant que document modal, attaché à la

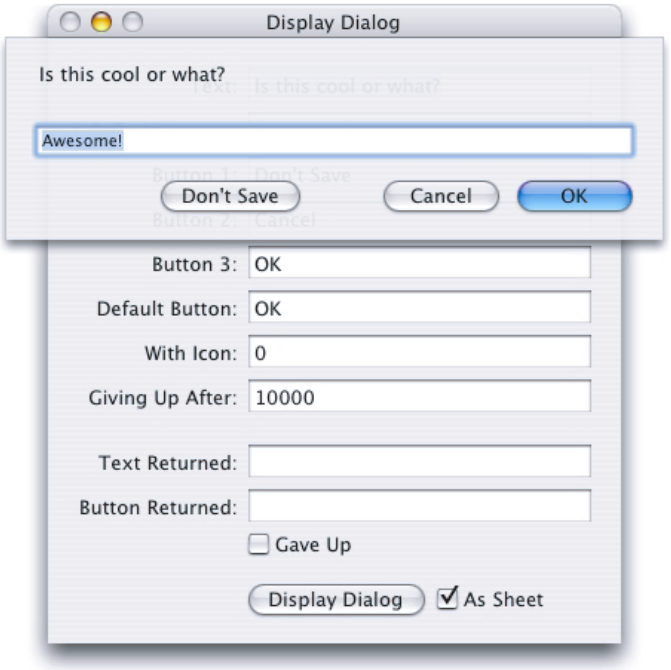

FIG. 9.6 - *Un dialogue "feuille" affiché par la commande Display Dialog*

fenêtre spécifiée

`[buttons]` *list*
La liste des noms des boutons, au maximum trois noms

`[default answer]` *Unicode text*
La réponse par défaut

`[default button]` *n'importe*
Le nom ou le numéro du bouton par défaut

`[giving up after]` *integer*
Le nombre de secondes à attendre avant que le dialogue ne se ferme automatiquement

`[with icon]` *n'importe*
Une chaîne de caractères ou un nombre entier qui spécifie l'icone à afficher ; une chaîne de caractères devra spécifier dans le projet un fichier TIFF (sans l'extension `.tiff`) contenant une image ; pour le nombre entier, vous pouvez utiliser les valeurs 0, 1 et 2 ; voir la section "Exemples" ci-dessous pour plus d'informations

Résultats

dialog reply

Un objet dialog reply (page 518) contenant les informations sur le dialogue renvoyé. Lorsque le dialogue est affiché en tant que "feuille" (attaché à une fenêtre), il n'y a pas de résultat immédiat, et vous devrez installer un gestionnaire dialog ended (page 552) pour répondre au dialogue renvoyé

Exemples

L'exemple suivant affiche un dialogue avec un texte et trois boutons. Il spécifie un délai (`giving up after 10`), un bouton par défaut (`default button "Goodbye"`) et un type d'icone (`with icon 0`). L'exemple affiche alors un second dialogue pour montrer quel bouton a été cliqué :

```
on clicked theObject
  set theReply to display dialog "Please click a button."
    buttons {"Hello", "Goodbye", "OK"} default button "Goodbye"
    giving up after 10 with icon 0
  display dialog button returned of theReply
end clicked
```

Dans cet exemple, le contrôle ne revient pas à la seconde instruction `display dialog` tant que l'utilisateur n'a pas cliqué sur un bouton (ou appuyé sur la touche Retour) ou que les 10 secondes imparties pour répondre ne sont pas écoulées.

Vous pouvez afficher une icone définie dans l'application en spécifiant le fichier TIFF de cet icone. Par exemple, si le fichier se nomme `myIcon.tiff` et que vous l'avez mis dans votre projet Xcode, vous pouvez remplacer `with icon 0` par `with icon myIcon` dans le gestionnaire Clicked ci-dessus.

Les constantes AppleScript `stop`, `note` et `caution` ne fonctionneront pas si vous vous en servez pour spécifier l'icone dans la commande Display Dialog de votre application. Utilisez plutôt les valeurs 0, 1 et 2 à la place. AppleScript Studio s'appuie sur les icones fournies par Cocoa pour ces valeurs. Dans Mac OS X version 10.2, transmettre 0 affichera une image avec un point d'exclamation, tandis que 1 ou 2 afficheront l'icone générique d'une application.

Vous pouvez, toutefois, utiliser les constantes `stop`, `note` et `caution` pour afficher un dialogue pour une autre application qu'une application

AppleScript Studio, à l'intérieur d'un bloc `tell application`, comme dans l'exemple suivant où le Finder est destinataire du dialogue :

```
on clicked theObject
  tell application "Finder"
    set theReply to display dialog "Please click a button."
      buttons {"Hello", "Goodbye", "OK"} default button "Goodbye"
      giving up after 10 with icon stop
    display dialog button returned of theReply
  end tell
end clicked
```

Voir la section "Discussion" pour plus d'informations sur la manière d'afficher un dialogue comme une "feuille" (attaché à une fenêtre). Pour un exemple plus complexe, voir l'application "Display Dialog" distribuée avec AppleScript Studio, ainsi que la section "Exemples" de la classe dialog reply (page 518).

Discussion

Lorsque vous affichez un dialogue indépendant, l'exécution du script s'arrête jusqu'à ce que le dialogue soit renvoyé, l'exécution reprend alors à l'instruction suivant immédiatement l'instruction `display dialog`.

Lorsque vous affichez un dialogue en tant que "feuille" (attaché à une fenêtre), l'exécution du script continue, et l'instruction suivante est exécutée immédiatement. Cela peut conduire à une certaine confusion. Pour reprendre le contrôle lorsque l'utilisateur a renvoyé le dialogue, vous installerez un gestionnaire dialog ended (page 552), comme le montre l'application "Display Dialog" distribuée avec AppleScript Studio.

Lorsque vous affichez un dialogue indépendant, si l'utilisateur appuie sur le bouton "Annuler" la commande Display Dialog génèrera une erreur "Annulé par l'utilisateur", aussi votre script devra utiliser un bloc `try, on error` (également montré dans l'application "Display Dialog") pour gérer cette erreur.

Lorsque vous affichez un dialogue attaché à une fenêtre, votre gestionnaire Dialog Ended peut traiter l'annulation comme un autre bouton.

Version

Depuis la version 1.1 d'AppleScript Studio, vous pouvez transmettre un nombre à la commande display dialog (page 543) sans avoir à le mettre au format string, comme dans l'instruction suivante (où `amount` et `rate` ont été précédemment définis comme étant des valeurs numériques) :

```
display dialog amount * rate
```

Pour un autre exemple, l'instruction suivante affiche un dialogue minimal (avec seulement les boutons "Annuler" et "Ok") contenant le chiffre "10" et se fermant tout seul au bout de 3 secondes si l'utilisateur ne répond pas :

```
display dialog 10 giving up after 3
```

display panel

Affiche le panel spécifié.

Non recommandée depuis la version 1.2 d'AppleScript Studio. Il est préférable d'utiliser la commande display (page 537) à la place. Les paramètres optionnels permettent de contrôler l'affichage de la fenêtre du panel.

Syntaxe

`display panel`	*reference*	obligatoire
`[afterwards calling]`	*n'importe*	facultatif
`[attached to]`	*window*	facultatif
`[for file types]`	*list*	facultatif
`[in directory]`	*Unicode text*	facultatif
`[with file name]`	*Unicode text*	facultatif

Paramètres

reference
: La référence de la fenêtre à afficher

`[afterwards calling]` *n'importe*
: Le script à lancer lorsque l'affichage est fini

`[attached to]` *window* (page 76)
: La fenêtre à laquelle attacher la fenêtre du panel

`[for file types]` *list*
: La liste d'extensions de fichiers qui sont autorisées (pour les panels d'enregistrement ou d'ouverture), comme “rtf” ou “txt” ; non inclus le point marquant l'extension

`[in directory]` *Unicode text*
: Le répertoire de départ (pour les panels d'enregistrement ou d'ouverture)

`[with file name]` *Unicode text*
: Le nom de ficher par défaut (pour les panels d'enregistrement ou d'ouverture)

Résultats

integer
: Une valeur entière représentant le bouton qui a renvoyé le panel ; une valeur 0 indique que le panel a été annulé ; 1 indique qu'il a été accepté

Exemples

Pour des exemples montrant comment charger et afficher un panel, voir les sections “Exemples” des gestionnaires panel ended (page 553) et load nib (page 106).

Version

Depuis la version 1.2 d'AppleScript Studio, cette commande n'est plus recommandée, utilisez plutôt à la place la commande display (page 537).

`load panel`

Non recommandée depuis la version 1.1 d'AppleScript Studio. Charge le panel spécifié depuis le fichier .nib spécifié.

Syntaxe

`load panel`	*reference*	obligatoire
`[from nib]`	*Unicode text*	facultatif

Paramètres

reference

La référence de la fenêtre du panel à charger

`[from nib]` *Unicode text*

Le nom du ficher .nib à partir duquel doit être obtenue la fenêtre du panel

Exemples

Pour utiliser cette commande, vous insérerez la fenêtre du panel dans votre fichier .nib et lui fournirez un titre dans la fenêtre “Attributes” de la fenêtre Info (ce n’est pas son nom AppleScript, le nom AppleScript est fourni dans le panneau “AppleScript”). Vous chargerez alors le panel dans le script en spécifiant le titre de la fenêtre. Par exemple, étant donné une fenêtre avec comme titre “myWindow” dans un fichier .nib “myNib”, vous pourrez alors charger une instance de cette fenêtre pour l’utiliser comme panel avec l’instruction suivante :

```
load panel "myWindow" from nib "myNib"
```

Version

Depuis la version 1.2 d’AppleScript Studio, cette commande n’est plus recommandée, utilisez plutôt la commande load nib (page 106) à la place.

Chapitre 3

Events

Les objets basés sur les classes de la suite Panel supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

alert ended 551
dialog ended 552
panel ended 553

`alert ended`

Appelé après qu'une alerte soit finie si le panel de celle-ci a été affiché comme étant attaché à une fenêtre.

Lorsque vous affichez une alerte en tant qu'application modale, votre script devra attendre que l'utilisateur renvoie l'alerte. Si vous l'affichez en tant que document modal en spécifiant le paramètre `attached to`, l'exécution du script continue. Pour exécuter toute action lorsque l'utilisateur renvoie l'alerte, vous devrez fournir un gestionnaire Alert Ended.

Syntaxe

`alert ended`	*reference*	obligatoire
`[with reply]`	*alert reply*	facultatif

Paramètres

reference
La référence de l'objet window (page 76) auquel a été attaché le panel

`[with reply]` *alert reply*
La réponse retournée par l'alerte

Exemples

Lorsque vous connectez dans Interface Builder un gestionnaire Alert Ended à un objet window (page 76), AppleScript Studio ajoute automatiquement au script désigné, un gestionnaire vierge identique à celui qui suit. Le gestionnaire suivant est extrait de l'application "Display Alert" distribuée avec AppleScript Studio. Ce gestionnaire obtient, à partir du paramètre `with reply`, le nom du bouton retourné et l'utilise pour régler le contenu d'un champ texte nommé "button returned".

```
on alert ended theObject with reply theReply
  set contents of text field "button returned" of window "main"
    to button returned of the reply
end alert ended
```

dialog ended

Appelé après qu'un dialogue soit fini si ce dialogue a été attaché à une fenêtre.

Lorsque vous affichez un dialogue en tant qu'application modale, votre script devra attendre que l'utilisateur ait renvoyé le dialogue. Si vous l'affichez en tant que document modal en spécifiant le paramètre `attached to`, l'exécution du script continue. Vous devrez fournir un gestionnaire Dialog Ended afin de pouvoir exécuter toute action lorsque l'utilisateur renverra le dialogue,.

Syntaxe

`dialog ended`	*reference*	obligatoire
`[with reply]`	*dialog reply*	facultatif

Paramètres

reference
La référence du panel pour lequel le gestionnaire est appelé

`[with reply]` *dialog reply*
La réponse retournée depuis le dialogue

Exemples

Pour un exemple de gestionnaire Dialog Ended, voir la section "Exemples" de la commande dialog reply (page 518).

`panel ended`

Appelé après qu'un panel ait fini, si ce panel a été affiché comme étant attaché à une fenêtre.

Lorsque vous affichez un panel en tant qu'application modale, votre script devra attendre que l'utilisateur ait renvoyé le panel. Si vous l'affichez en tant que document modal en spécifiant le paramètre `attached to`, l'exécution du script continue. Vous devrez fournir un gestionnaire Panel Ended afin de pouvoir exécuter toute action lorsque l'utilisateur renverra le panel,.

Le gestionnaire Panel Ended est appelé dans le script qui est connecté à la fenêtre à laquelle est connecté l'objet open-panel (page 522) ou save-panel (page 529). Par exemple, si vos gestionnaires d'Events gérant les fenêtres sont connectés dans un script nommé `Window.applescript` et que votre script affiche un panel avec une instruction comme celle qui suit `display open panel attached to window "main"`, alors le gestionnaire Panel Ended devra aussi être connecté à `Window.applescript`. Ce gestionnaire sera appelé lorsque le panel sera renvoyé.

Syntaxe

`panel ended`	*reference*	obligatoire
`[with result]`	*n'importe*	facultatif

Paramètres

reference
La référence du panel pour lequel le gestionnaire est appelé

`[with result]` *n'importe*
Le résultat retourné par le panel

Exemples

Le gestionnaire Panel Ended suivant est extrait de l'application "Save Panel", distribuée depuis la version 1.1 d'AppleScript Studio. Vous connecterez ce gestionnaire à une fenêtre dans Interface Builder. Alors votre application appellera la commande display panel (page 547) en utilisant le paramètre `attached to window`, pour attacher le panel à la fenêtre avec le gestionnaire Panel Ended. Ce gestionnaire sera appelé lorsque le panel sera renvoyé.

```
on panel ended theObject with result withResult
  if withResult is 1 then
    set contents of text field "path name" of window "main"
      to path name of save panel
  else
    set contents of text field "path name" of window "main" to ""
  end if
end panel ended
```

L'application "Save Panel" utilise l'instruction suivante pour afficher le panel attaché à la fenêtre qui utilise le gestionnaire Panel Ended montré ci-dessus. Les paramètres `in directory` et `with file name` sont facultatifs.

```
display save panel in directory theDirectory
  with file name theFileName attached to window of theObject
```

Pour plus d'informations, voir la section "Exemples" de la commande display (page 537).

Version

L'application "Save Panel" a été distribuée pour la première fois avec la version 1.1 d'AppleScript Studio.

Dixième partie

Plugin Suite

Cette partie décrit la terminologie de la suite Plugin d'AppleScript Studio.

Depuis la version 1.3 d'AppleScript Studio, distribuée pour la première fois avec Mac OS X 10.3, Xcode fournit un nouveau gabarit pour la création de plugin AppleScript à destination de Xcode. C'est à dire que vous pouvez utiliser AppleScript Studio pour créer des plugins qui ajoutent des caractéristiques à Xcode lui-même. La suite Plugin fournit la terminologie à utiliser avec les plugins de ce type.

Les classes et Events de la suite Plugin sont décrits dans les sections suivantes :

Classes 559
Events 561

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Plugin contient la classe suivante :

plugin 559

plugin

Pluriel : plugins
Hérite de : item (page 60)
Classe Cocoa : ASKPlugin

Représente un plug-in.

Vous construirez des plug-ins en utilisant le modèle de projet Xcode “AppleScript Xcode Plugin”. Une fois que vous avez construit un plug-in, vous le placerez à un de ces emplacements, ainsi il sera chargé au prochain lancement de Xcode :

- /Library/Application Support/Apple/Developer Tools/Plug-Ins
- ~/Library/Application Support/Apple/Developer Tools/Plug-Ins
- /Network/Library/Application Support/Apple/Developer Tools/Plug-Ins

Un objet plugin a uniquement accès en lecture aux propriétés de sa classe parent.

Exemples

La raison principale de la création d'un plug-in AppleScript Xcode est d'ajouter de nouvelles fonctionnalités à Xcode en tirant avantage de sa scriptabilité. Cela peut être utile, car Xcode supporte une terminologie de scripting robuste, et il est ainsi possible de fournir, par l'intermédiaire de l'interface graphique de Xcode, des caractéristiques généralement non disponibles.

Pour un exemple montrant comment créer un menu et un élément de menu à Xcode, voir la section "Exemples" de la commande plugin loaded (page 561).

Version

La classe Plugin fut ajoutée dans la version 1.3 d'AppleScript Studio.

Chapitre 2

Events

Les objets basés sur les classes de la suite Plugin supportent les gestionnaires d'Events suivants (un **Event** est une action, généralement générée par l'interaction avec l'interface utilisateur, provoquant l'appel du gestionnaire approprié devant être exécuté). Pour déterminer les Events supportés par une classe, se référer à sa description.

plugin loaded 561

`plugin loaded`

Appelé lorsque le plug-in pour lequel le gestionnaire est attaché est chargé.

Vous utiliserez généralement ce gestionnaire pour créer n'importe quel élément d'interface que vous ajoutez à Xcode (comme un menu ou un élément de menu) et pour créer les scripts gérant les actions invoquées par ces éléments d'interface.

Syntaxe

`plugin loaded` *reference* obligatoire

Paramètres

reference

La référence de l'objet dont le gestionnaire Plugin loaded est appelé

Exemples

Cet exemple décrit comment créer un plug-in et l'utiliser pour ajouter un menu et un élément de menu à Xcode.

Pour créer un projet AppleScript plug-in dans Xcode, suivez ces étapes :

1. Choisissez File ¿ New Project.
2. Dans la fenêtre "New Project Assistant", faites défiler les choix jusqu'à la section "Standard Apple Plug-ins".
3. Sélectionnez "AppleScript Xcode Plugin".

Pour connecter le gestionnaire à l'application, vous suivrez ces étapes :

1. Sélectionnez le projet dans la liste "Groups & Files" de Xcode. Les fichiers du projet devraient alors être visibles dans la présentation détaillée.
2. Double-cliquez le fichier .nib pour l'ouvrir dans Interface Builder.
3. Dans Interface Builder, sélectionnez "File's Owner" dans la fenêtre Nib.
4. Ouvrez la fenêtre Info (soit avec Cmd + Maj. + I, soit avec le menu Tools ¿ Show Info).
5. Affichez le panneau AppleScript (appuyez sur Cmd + 7 ou utilisez le menu déroulant).
6. Ouvrez le groupe "Plugin" et sélectionnez le gestionnaire Plugin Loaded.
7. Sélectionnez le fichier script, puis cliquez sur le bouton "Edit Script". Cela devrait vous ramener dans l'application Xcode, avec le fichier script ouvert et contenant un gestionnaire Plugin Loaded vierge.

Le listing ci-dessous fournit un exemple de code que vous pouvez mettre dans un gestionnaire Plugin Loaded. Cet exemple ajoute un menu "Plug-ins" dans Xcode et un élément de menu "Display Dialog" dans ce menu. Il définit aussi un script et attache le script à l'élément de menu. Le script affiche un dialogue.

Rappelez-vous qu'après la construction de votre plug-in, vous devrez le placer dans un des trois emplacement décrits dans la classe plugin (page 559) afin qu'il soit chargé par Xcode lors de son prochain lancement.

```
on plugin loaded theBundle
  -- Make a new script
```

```
  script pluginScript
    property someVariable : "Display Dialog"

    on choose menu item theObject
      display dialog someVariable
    end choose menu item
  end script

  -- Make a new plugin menu
  set pluginMenu to make new menu at end of menus of main menu with  properties {title:"

  -- Make a new Display Dialog menu item in the plugin menu set pluginItem to make new m

  -- Set the script of the new menu item
  set script of pluginItem to pluginScript

end plugin loaded
```

Version

Le gestionnaire Plugin Loaded fut ajouté dans la version 1.3 d'AppleScript Studio.

Onzième partie

Text View Suite

Cette partie décrit la terminologie de la suite Text View d'AppleScript Studio.

La suite Text View définit deux classes pour l'affichage et la manipulation du texte. La classe text (page 569) hérite de la classe view (page 229) et la classe text view (page 569) hérite de la classe text (page 569). Les classes de la suite Text View sont décrites dans le chapitre suivant :

Classes 569

Le chapitre "Énumérations" (page 173) de "Application Suite" (page 27) détaille les différentes constantes utilisées dans cette suite.

Chapitre 1

Classes

La suite Text View contient les classes suivantes :

text 569
text view 569

text

Pluriel : text
Hérite de : view (page 229)
Classe Cocoa : NSText

Non disponible depuis la version 1.3 d'AppleScript Studio.

Attention
Depuis la version 1.3 d'AppleScript Studio, la classe Text est fusionnée avec la classe text view (page 569). Les propriétés précédemment accessibles dans la classe Text sont dorénavant dans la classe Text view. Voir la section "Version" de la classe Text View pour la liste complète des propriétés fusionnées.

text view

Pluriel : text views
Hérite de : text (page 569)
Classe Cocoa : NSTextView

Fournit plusieurs caractéristiques pour l'affichage et la manipulation du texte disposé dans un espace défini.

Attention
Depuis la version 1.3 d'AppleScript Studio, la classe Text est fusionnée avec la classe text view (page 569). Les propriétés précédemment accessibles dans la classe Text sont dorénavant dans la classe Text view. Voir la section "Version" de la classe Text View pour la liste complète des propriétés fusionnées.

L'illustration 11.1 montre une fenêtre contenant un objet text view.

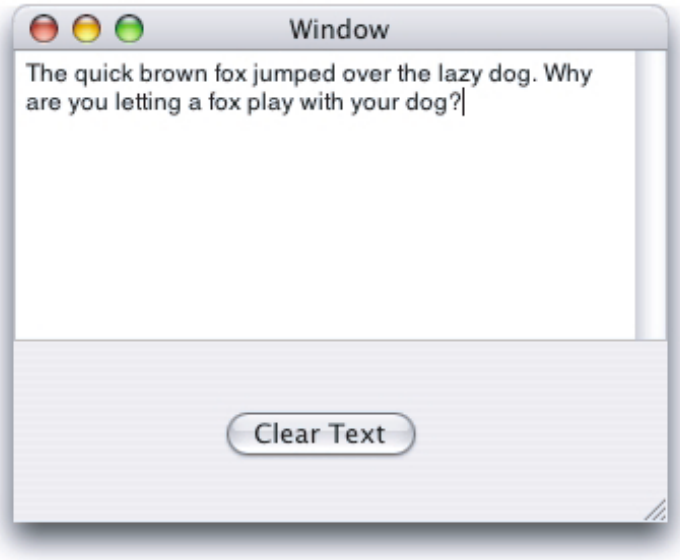

FIG. 11.1 - *Un text view contenant du texte*

Vous trouverez l'objet text view dans le panneau "Cocoa-Text" d'Interface Builder. Vous pouvez régler la plupart des attributs des objets text view dans la fenêtre Info d'Interface Builder.

Pour un aperçu du système de gestion du texte de Cocoa, voir Text System Architecture dans la documentation Cocoa.

Propriétés des objets de la classe Text View

En plus des propriétés qu'il hérite de la classe view (page 229), un objet text view possède ces propriétés :

alignment

Accès : lecture / écriture
Classe : *une des constantes* de text alignment (page 189)

L'alignement du texte

allows undo

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le text view autorise l'annulation ?

background color

Accès : lecture / écriture
Classe : *RGB color*
La couleur de fond de la view ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur verte peut être représentée par {0, 65535, 0}

content

Accès : lecture / écriture
Classe : *Unicode text*
Le contenu de la view ; synonyme de *contents*

contents

Accès : lecture / écriture
Classe : *Unicode text*
Le contenu de la view ; synonyme de *content*

draws background

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la view dessine son fond ?

editable

Accès : lecture / écriture
Classe : *boolean*
La view est-elle éditable ?

field editor

Accès : lecture / écriture
Classe : *boolean*
Est-ce un champ éditeur ? Un champ éditeur est utilisé par les objets supportant du texte ; par exemple, un objet text field (page 325) utilise le champ éditeur de sa fenêtre pour afficher et manipuler le texte ; le champ éditeur peut être partagé par n'importe quel nombre

d'objets et aussi, son état peut être constamment sur le point de se modifier ; pour plus d'informations, voir la description de la méthode `fieldEditor:forObject:` de la classe NSWindow

font

Accès : lecture / écriture
Classe : *font* (page 54)
La police de la view

horizontally resizable

Accès : lecture / écriture
Classe : *boolean*
La view est-elle redimensionnable horizontalement ?

imports graphics

Accès : lecture / écriture
Classe : *boolean*
Faut-il que la view importe des graphiques ?

maximum size

Accès : lecture / écriture
Classe : *point*
La taille maximale de la view ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

minimum size

Accès : lecture / écriture
Classe : *point*
La taille minimale de la view ; la taille est exprimée sous forme d'une liste de deux nombres {horizontal, vertical} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

rich text

Accès : lecture / écriture
Classe : *boolean*
Le texte supporte-t-il le format RTF (Rich Text Format) ? Par défaut, cette propriété vaut `true`

ruler visible

Accès : lecture / écriture
Classe : *boolean*
La règle est-elle visible ?

selectable

Accès : lecture / écriture
Classe : *boolean*
La view est-elle sélectionnable ?

smart insert delete enabled

Accès : lecture / écriture
Classe : *boolean*
Est-ce que la "carte" d'insertion et de suppression de texte est activée ? Par défaut, cette propriété vaut `true` ; contrôle si la view insère ou supprime les espaces autour des mots insérés ou supprimés afin de préserver un espacement et une ponctuation propres

spell checking enabled

Accès : lecture / écriture
Classe : *boolean*
Est-ce que la correction orthographique du text view est activée ?

text color

Accès : lecture / écriture
Classe : *RGB color*
La couleur du texte ; une liste de trois nombres entiers contenant les valeurs de chaque composant de la couleur ; par exemple, la couleur verte peut être représentée par {0, 65535, 0}

text container inset

Accès : lecture / écriture
Classe : *point*
La quantité d'espace libre que la view laisse autour du container texte associé ; un container texte représente l'endroit où le texte est arrangé ; le cartouche est exprimé sous forme d'une liste de deux nombres {largeur, hauteur} ; par défaut, cette propriété vaut {0, 0}

text container origin

Accès : lecture uniquement
Classe : *point*

L'origine du container texte à l'intérieur du text view, laquelle est calculée à partir des frontières rectangulaires de la view, du cartouche et du rectangle utilisé par le container ; un container texte représente l'endroit où le texte est arrangé ; l'origine est exprimée sous forme d'une liste de deux nombres {gauche, droite} ; voir la propriété *bounds* de la classe window (page 76) pour plus d'informations sur le système des coordonnées

uses font panel

Accès : lecture / écriture
Classe : *boolean*
La view peut-elle utiliser le panel "Polices" ?

uses ruler

Accès : lecture / écriture
Classe : *boolean*
Faut-il que le text view utilise des règles ?

vertically resizable

Accès : lecture / écriture
Classe : *boolean*
La view est-elle redimensionnable verticalement ?

Éléments des objets de la classe Text View

En plus des éléments qu'il hérite de la classe view (page 229), un objet text view peut contenir les éléments listés ci-dessous. Votre script peut accéder à la plupart de ces éléments avec les formes-clés décrites dans "Les formes-clés standards" (page 15).

`text` (page 569)
spécifier par : "Les formes-clés standards" (page 15)
Le texte de la view

Events supportés par les objets de la classe Text View

Un objet text view supporte les gestionnaires répondant aux Events suivants :

Glisser-Déposer

`conclude drop` (page 481)

`drag` (page 483)
`drag entered` (page 483)
`drag exited` (page 484)
`drag updated` (page 485)
`drop` (page 486)
`prepare drop` (page 488)

Édition

`begin editing` (page 346)
`changed` (page 348)
`end editing` (page 350)
`should begin editing` (page 353)
`should end editing` (page 354)

Clavier

`keyboard up` (page 134)

Souris

`mouse entered` (page 139)
`mouse exited` (page 139)
`scroll wheel` (page 151)

Nib

`awake from nib` (page 123)

View

`bounds changed` (page 243)

Exemples

L'instruction suivante, extraite de l'application "Open Panel" distribuée depuis la version 1.1 d'AppleScript Studio, montre comment régler le texte d'un objet text view. L'instruction nettoie le text view en réglant son texte sur une chaîne vide.

```
set contents of text view "path names" of scroll view "path names" of window
"main" to ""
```

Vous pouvez utiliser le script suivant dans l'application Éditeur de Scripts pour régler la couleur du texte d'un text view sur la couleur verte. Ce script peut tout à fait être repris dans le script d'une application AppleScript Studio, bien que vous n'aurez pas besoin de l'insérer dans un bloc `tell application`.

```
tell application "myTextViewApp"
  tell text view "text" of scroll view "scroller" of window "main"
    set text color to {0, 65535, 0}
  end tell
end tell
```

La terminologie pour la gestion du texte peut être un peu confuse. La suite Text View d'AppleScript Studio définit la classe text view (page 569), laquelle (avant la version 1.3 d'AppleScript Studio) hérite de la classe text (page 569) — depuis la version 1.3, les propriétés de la classe text (page 569) sont fusionnées avec la classe Text View.

En plus, Cocoa définit la suite Texte, laquelle définit les classes comme `character`, `paragraph`, `text` et `word`, lesquelles à leur tour ont des éléments comme `character`, `paragraph` et `word`, et des propriétés comme `color`, `font` et `size`. La suite Text est une suite par défaut qui est accessible à toutes les applications Cocoa qui supportent le scripting.

Pour davantage compliquer l'affaire, les classes AppleScript comme `string` et `Unicode text` ont des éléments `character`, `paragraph`, `text` et `word`. En plus, si une classe et une propriété (comme `text`) ont le même nom, utilisez leur nom à l'intérieur d'une instruction `tell` prend par défaut la classe.

Le script suivant montre diverses opérations que vous pouvez exécuter sur le texte d'un text view dans une fenêtre. Un objet text view est automatiquement encadré par un objet scroll view (page 213), aussi le script accède à l'objet text view par l'intermédiaire de l'objet scroll view. Ce script fut testé avec la version 1.2 d'AppleScript Studio dans l'application Éditeur de Scripts, mais vous pouvez l'utiliser dans le script d'une application AppleScript Studio (sans avoir besoin du bloc `tell application`). Ce script pourrait inclure des instructions incompatibles avec les versions antérieures d'AppleScript Studio.

Pour cet exemple, le texte du text view fut "This is the only sentence". Notez que vous obtiendrez une évaluation des erreurs pour certaines lignes de ce script s'il n'y a aucun texte dans le text view, ou, par exemple, moins

de 7 caractères (à cause de l'instruction `character 7 of text of text view 1` qui pourrait générer une erreur).

```
tell application "simple"
  tell window 1
    tell scroll view 1
      class of text of text view 1 -- text
      word 1 of text of text view 1 -- result: "This"
      set myTextObject to text of text view 1
      -- result: "This is the only sentence."
      class of myTextObject
      -- result: Unicode text
      -- In Studio version 1.1, the result is string.
      -- Next line generates error because Unicode text doesn't
      -- have a color property
      --color of myTextObject
      -- result: Can't get color of "This is the only sentence."
      word 1 of myTextObject -- result: "This"
      character 7 of myTextObject -- result: "s"
      character 1 of word 1 of myTextObject -- result: "T"
      set myText to contents of text view 1
      -- result: "This is the only sentence."
      class of myText -- result: Unicode text
      word 3 of myText -- result: "the"
      character 13 of myText -- result:"o"
      editable of text view 1 -- result: true (inherited from text)
      background color of text view 1 -- result: {65535, 65535, 65535}
      set myTextRef to a reference to (text of text view 1)
      -- result: every text of text view 1 of scroll view 1
      -- of window 1 of application "simple"
      class of myTextRef -- result: text
      color of myTextRef -- result: {0, 0, 0}
      --font of myTextRef -- NSCannotCreateScriptCommandError
      contents of myTextRef
      -- result: "This is the only sentence."
      size of myTextRef -- result: 12.0
      word 1 of myTextRef -- result: "This"
      color of word 1 of myTextRef -- result: {0, 0, 0}
      set color of word 1 of myTextRef to {65535, 0, 0}
      -- result: color of first worth ("This") is red
```

```
    end tell
  end tell
end tell
```

Voir la section “Exemples” de la commande scroll (page 339) pour plus d’informations sur comment faire défiler le texte d’un objet text view.

Voir la section “Exemples” de la classe view (page 229) pour plus d’informations sur comment faire tourner le texte dans un objet text view.

Version

Depuis la version 1.3 d’AppleScript Studio, la classe text (page 569) a été fusionnée avec la classe Text View. Les propriétés suivantes, précédemment dans la classe Text, sont maintenant accessibles par l’intermédiaire de la classe Text View : *alignment*, *background color*, *content*, *contents*, *draws background*, *editable*, *field editor*, *font*, *horizontally resizable*, *imports graphics*, *maximum size*, *minimum size*, *rich text*, *selectable*, *text color*, *uses font panel* et *vertically resizable*.

Depuis la version 1.2 d’AppleScript Studio, un script peut dire `word 1 of text view 1` au lieu de `word 1 of text of text view 1` (le `of text` est présumé), bien que la seconde version fonctionne toujours.

Le support des Events de Glisser-Déposer est apparu avec la version 1.2 d’AppleScript Studio. Voir la suite “Drag and Drop Suite” (page 475) pour plus de détails. En particulier, la description du gestionnaire conclude drop (page 481) donne des informations le support du Glisser-Déposer des objets text view (page 569) et text field (page 325).

La propriété *content* est apparue avec la version 1.2 d’AppleScript Studio. Vous pouvez utiliser au choix *content* et *contents*, sauf à l’intérieur d’un gestionnaire d’Events, `contents of theObject` retournant une référence à l’objet plutôt que son contenu courant. Pour obtenir dans un gestionnaire d’Events, le contenu d’un objet (comme le texte contenu dans un text field (page 325)), vous pouvez utiliser soit `contents of contents of theObject`, soit `content of theObject`.

Pour un exemple de script montrant la différence entre `content` et `contents`, voir la section “Version” de la classe control (page 280).

Voir la section “Exemples” de la commande scroll (page 339) pour plus d’informations sur la manière de faire défiler le texte dans un text view (page 569).

Douzième partie

Annexes

Annexe A

Les applications distribuées avec AppleScript Studio

À l'appui d'AppleScript Studio version 1.3, Apple fournit un certain nombre d'exemples d'applications AppleScript Studio. Ces exemples sont situés dans le répertoire `/Developer/Examples/AppleScript Studio/` et sont tous livrés à l'état de projet — c'est à dire que vous devrez d'abord les compiler dans l'application Xcode avant de pouvoir les essayer en état de fonctionnement normal. De plus, ces exemples étant livrés à l'état de projet, vous aurez accès à leurs sources complets, interface graphique et code AppleScript Studio. Ces exemples peuvent tout à fait être modifiés, améliorés, bref, vous pouvez les manipuler dans tous les sens et les recompiler, si cela avait déjà été fait, afin de voir vos changements.

Au total, il y a 33 exemples d'applications livrés, couvrant si possible, l'ensemble des caractéristiques d'AppleScript Studio version 1.3. Certains modèles nécessitent une connexion internet pour fonctionner correctement (cas de Currency Converter (SOAP), Daily Dilbert, SOAP Talk entre autres) car ils ont besoin de récupérer des infos sur le net, les autres fonctionnent en mode local.

Vous trouverez dans les sections de cette annexe, un rapide descriptif de certains modèles, mais pas des 33 disponibles, uniquement de 4. Cette annexe n'a pas pour but de vous expliquer en détails leur fonctionnement ou le pourquoi du code des scripts, mais de vous montrer ce que vous pouvez obtenir avec la technologie AppleScript Studio basée sur le langage de scripting AppleScript, pour résumer avec un Mac et les outils développeurs mis gratuitement à disposition par Apple (oui, je sais 300 Mo, ça ne se charge pas en cinq minutes, surtout en RTC, mais ils sont tout de même mis

gratuitement en ligne, tout le monde ne le fait pas). :-)))))))))))

Notez que tous ces modèles sont en langue anglaise, il n'y a pas d'autre localisation disponible, donc si vous voulez une version française, il faudra la faire vous-même, vous pourrez ainsi vous faire la main. :-)))

Les modèles présentés sont :

Archive Maker 582
Unit Converter 583
Table Sort 584
Save Panel 586

Archive Maker

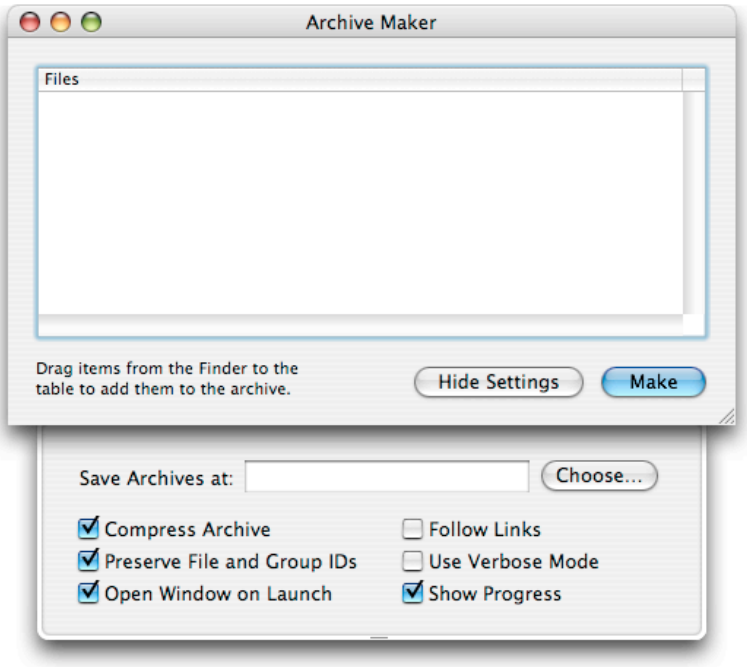

FIG. 12.1 - *La fenêtre de l'application "Archive Maker"*

Archive Maker est une interface graphique pour l'utilitaire d'archivage GNUtar permettant de créer des archives .tar ou .tar.gz pour des archives compressées. Cette application se sert de la commande AppleScript `Do Shell Script` permettant d'utiliser des commandes UNIX dans un script

AppleScript. Grâce à cette commande, vos scripts AppleScript peuvent utiliser les commandes UNIX sans avoir besoin de lancer l'application Terminal. AppleScript Studio est certainement la manière la plus simple et la plus rapide pour créer une interface graphique à un utilitaire UNIX.

Cette application utilise aussi, entre autres, le support du Glisser-Déposer, un tiroir dans lequel vous pouvez choisir le lieu d'enregistrement de votre archive, plus diverses autres options. Le bouton "Hide Settings" sert à fermer le tiroir, son intitulé change en fonction de l'état du tiroir — "Hide Settings" lorsqu'il est ouvert, "Show Settings" lorsqu'il est fermé. Mais aussi également un objet data source (page 386) pour stocker les noms des fichiers et le système des valeurs utilisateurs par défaut pour stocker les réglages de l'application de l'utilisateur courant.

Unit Converter

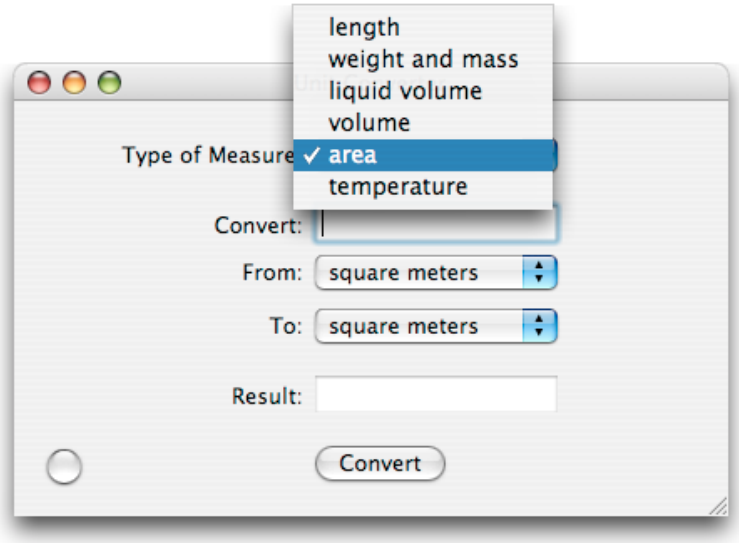

FIG. 12.2 - *La fenêtre de l'application "Unit Converter" avec les choix des types de mesure*

Unit Converter sert à convertir des unités de mesure. Le type de mesure se fait dans le menu déroulant "Type of Measure", les choix possibles sont : Length, Weight and Mass, Liquid Volume, Volume, Area et Temperature. Le type de mesure choisi conditionne les choix possibles dans les deux autres menus déroulants, "From" et "To". Par exemple, si vous optez pour le type "Temperature", vous pourrez convertir des degrés Fahrenheit

en degrés Celsius, si vous optez pour le type “Length”, vous pourrez convertir des kilomètres en yards. La mise à jour de ces deux menus déroulants est effectuée une fois le choix fait dans le type de mesure et elle est instantanée.

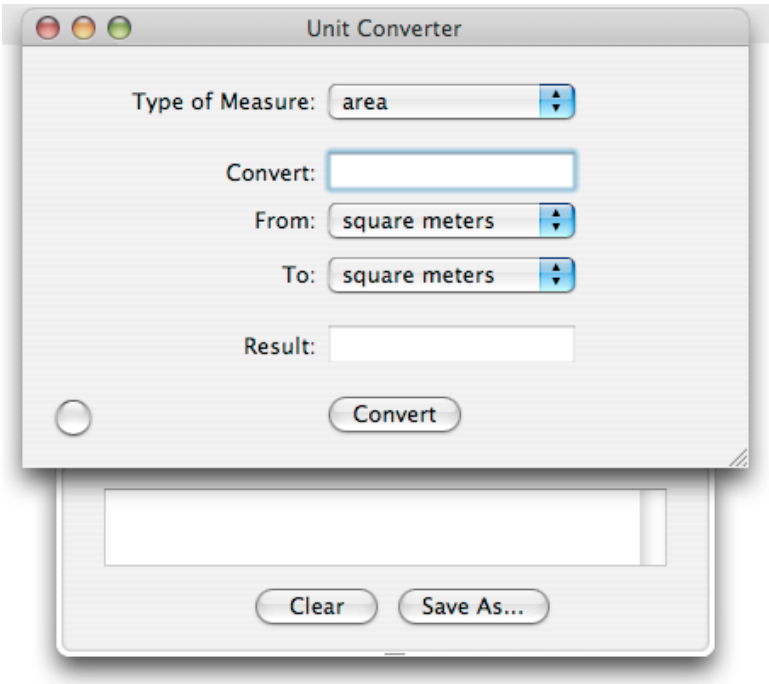

FIG. 12.3 - *L’application “Unit Converter” avec son tiroir ouvert*

Dans l’illustration ci-dessus, vous pouvez voir le tiroir qui s’ouvre sous la fenêtre lorsque vous cliquez sur le bouton rond situé dans le coin inférieur gauche. Ce tiroir récapitule toutes les conversions effectuées depuis le lancement de l’application et vous permet de les sauvegarder.

Table Sort

Table Sort sert à présenter des données dans un tableau, ici des informations sur des personnes. Cette application se sert d’un objet data source (page 386) pour stocker provisoirement les données des différentes personnes et alimenter le tableau en données. Pour garder de manière permanente ses données, d’un lancement à un autre, vous devrez récupérer les données de la data source et les enregistrer dans un fichier annexe sous une forme facilement exploitable pour le prochain lancement. Le format List est certainement le meilleur choix, mais chacun est libre de faire comme il veut.

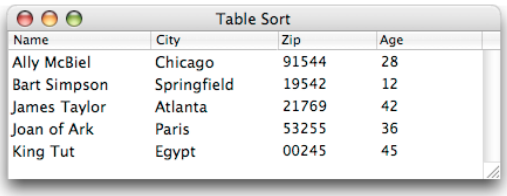

FIG. 12.4 - *L'application "Table Sort" avec la liste des individus*

En cliquant sur l'en-tête d'une colonne, comme dans l'illustration suivante, vous provoquerez le tri des données de cette colonne et la mise à jour complète du tableau. Cela est possible car un gestionnaire column clicked (page 430) est connecté à l'objet table view (page 403) dans Interface Builder. Le tri se fera de façon croissante ou décroissante à chaque nouveau clic sur l'en-tête.

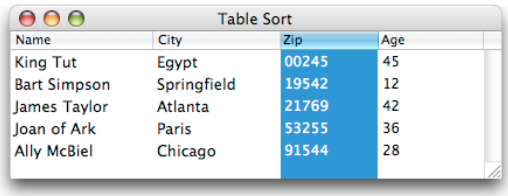

FIG. 12.5 - *La colonne "Zip" retriée suite au clic sur son en-tête*

Il est également possible de modifier le contenu du tableau, tout simplement en double-cliquant sur la donnée à modifier. La modification se fera à l'écran et également dans la data source de façon transparente. Il est tout à fait possible d'imposer un type pour les données saisies, par exemple, que des chiffres pour les données de la colonne "Zip". L'illustration suivante montre la sélection du contenu d'une cellule du tableau afin d'y apporter une modification, la sélection a été faite en double-cliquant avec la souris sur la cellule.

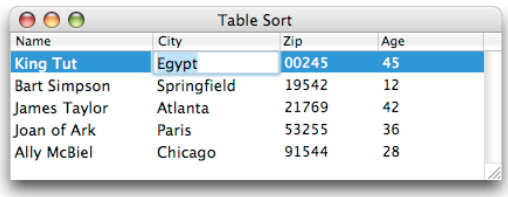

FIG. 12.6 - *Sélection du contenu de la colonne "City" avant modification*

Save Panel

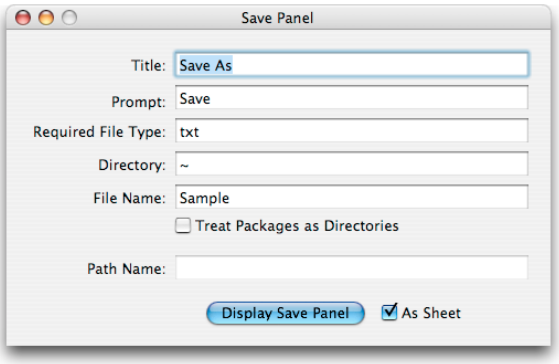

FIG. 12.7 - *La fenêtre de l'application "Save Panel" avec les différents choix possibles*

Save Panel sert uniquement à montrer l'utilisation du panel d'enregistrement dans une application AppleScript Studio. Vous pouvez régler la plupart des propriétés d'un objet open-panel (page 522) dans sa fenêtre, ainsi vous pouvez tester les différents réglages possibles et voir le résultat en cliquant sur le bouton "Display Save Panel".

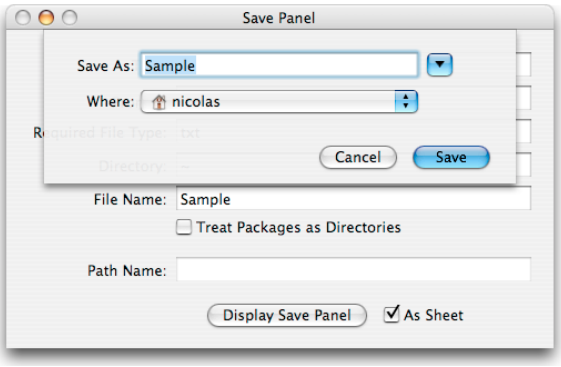

FIG. 12.8 - *Le panel d'enregistrement attaché à la fenêtre*

À côté du bouton "Display Save Panel", se trouve une case à cocher intitulé "As Sheet". Elle permet de choisir si le panel d'enregistrement doit être attaché ou non à la fenêtre. Si la case est cochée, le panel d'enregistrement sera attaché à la fenêtre et apparaîtra à l'écran en sortant de sous la barre de titre de la fenêtre, comme un store qui descend pour l'ouverture et qui remonte pour la fermeture. De plus, si votre fenêtre a une largeur inférieure à celle du panel d'enregistrement, vous aurez droit à l'effet génie, comme un

génie qui sort de sa lampe, idem pour la fermeture, effet garanti. :-)))

Si la case n'est pas cochée, le panel ne sera pas attaché à la fenêtre de l'application et sera affiché dans une fenêtre à part. Vous pourrez déplacer la fenêtre du panel, mais elle sera toujours au-dessus de celle de l'application, et vous ne pourrez pas modifier le contenu de l'application tant que vous n'aurez pas renvoyé le panel en cliquant, soit sur le bouton "Enregistrer", soit sur le bouton "Annuler".

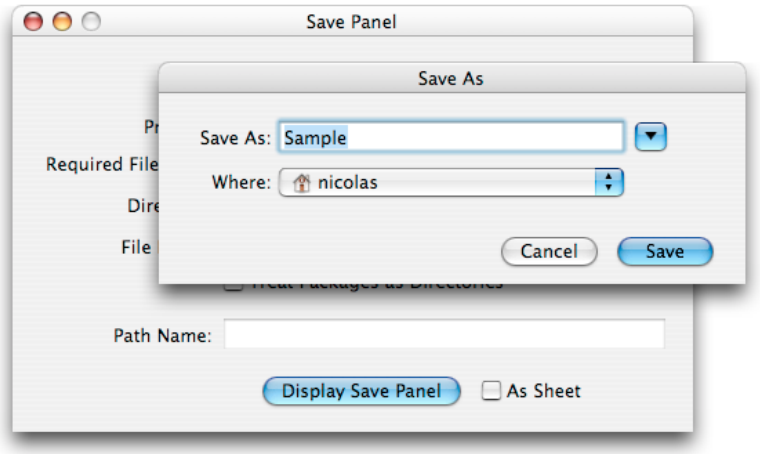

FIG. 12.9 - *Le panel d'enregistrement affiché dans une fenêtre à part*

Index

A

above bottom
constante *190*
above top
constante *190*
accepts arrow keys
propriété *364*
action
event *345*
action cell
classe *254*
activated
event *123*
active
propriété *31*, *272*
alert ended
event *551*
alert reply
classe *513*
alert return values
énumération *174*
alert type
énumération *174*
alignment
propriété *265*, *280*, *570*
allows branch selection
propriété *365*
allows column reordering
propriété *404*
allows column resizing
propriété *405*
allows column selection
propriété *405*
allows editing text attributes
propriété *265*, *325*
allows empty selection
propriété *290*, *365*, *405*
allows mixed state
propriété *255*, *265*
allows multiple selection
propriété *365*, *405*, *523*
allows undo
propriété *571*
alpha
propriété *516*
alpha value
propriété *78*
alphabetical
constante *188*
alternate image
propriété *255*, *261*, *371*
alternate increment value
propriété *316*
alternate return
constante *174*
alternate title
propriété *255*, *262*
animate
commande *333*
animation delay
propriété *307*
append
commande *417*

appkit defined type
constante *181*
application
classe *29*
application defined type
constante *181*
ascending
constante *187*
associated file name
propriété *78*
associated object
propriété *265*, *384*, *500*
at bottom
constante *190*
at top
constante *190*
auto completes
propriété *275*
auto display
propriété *78*
auto enables items
propriété *302*, *496*
auto repeat
propriété *321*
auto resizes
propriété *229*
auto resizes all columns to fit
propriété *405*
auto resizes outline column
propriété *393*
auto save expanded items
propriété *393*
auto save name
propriété *405*
auto save table columns
propriété *406*
auto scroll
propriété *290*
auto sizes cells
propriété *290*
awake from nib
event *123*

B

background
constante *188*
background color
propriété .. *78*, *202*, *214*, *290*, *326*, *330*, *406*, *571*
became key
event *127*
became main
event *128*
begin editing
event *346*
beginning frame
constante *183*
below bottom
constante *191*
below top
constante *191*
bezel border
constante *176*
bezel style
énumération *175*
propriété *255*, *262*
bezeled
propriété *265*, *307*, *326*
border rect
propriété *198*
border type
énumération *175*
propriété *198*, *214*
bordered
propriété .. *256*, *265*, *272*, *326*
bottom
constante *187*
bottom alignment
constante *183*

bottom edge
constante *186*
bottom left alignment
constante *183*
bottom right alignment
constante *183*
bottom tabs bezel border
constante *189*
bounds
propriété *79*, *230*
bounds changed
event *243*
bounds rotation
propriété *230*
box
élément *85*, *209*, *233*
classe *197*
box type
énumération *176*
propriété *198*
browser
élément *85*, *209*, *233*
classe *363*
browser cell
classe *370*
bundle
classe *38*
button
élément *85*, *209*, *233*
classe *254*
button cell
classe *261*
button frame
constante *184*
button returned
propriété *514*, *518*
button type
énumération *177*
propriété *256*, *262*

C

call method
commande *94*
can choose directories
propriété *523*
can choose files
propriété *523*
can draw
propriété *230*
can hide
propriété *79*
case insensitive
constante *187*
case sensitive
constante *187*
cell
élément *294*, *368*
classe *264*
propriété *280*
cell background color
propriété *291*
cell image position
énumération *177*
cell prototype
propriété *365*
cell size
propriété *265*, *291*
cell state value
énumération *178*
cell type
énumération *179*
propriété *266*
cell value
event *424*
center
commande *98*
center alignment
constante *183*

center text alignment
constante *189*
change cell value
event *426*
change item value
event *427*
changed
event *348*
characters
propriété *50*
child of item
event *428*
choose menu item
event *505*
circular bezel
constante *175*
clear tint
constante *180*
click count
propriété *51*
clicked
event *348*
clicked column
propriété *406*
clicked data column
propriété *406*
clicked data item
propriété *394*
clicked data row
propriété *406*
clicked row
propriété *407*
clip view
élément *85*, *210*, *233*
classe *202*
close drawer
commande *239*
close panel
commande *535*
closed
event *129*
cmyk mode
constante *179*
color
propriété *227*, *272*, *516*
color list mode
constante *179*
color mode
propriété *516*
color panel
propriété *31*
color panel mode
énumération *179*
color well
élément *85*, *210*, *233*
classe *271*
color wheel mode
constante *179*
color-panel
classe *514*
column clicked
event *430*
column moved
event *430*
column resized
event *431*
combo box
élément *85*, *210*, *233*
classe *274*
combo box item
élément *276*
classe *279*
command key down
propriété *51*
conclude drop
event *481*
content
propriété ... *46*, *65*, *266*, *280*, *307*, *373*, *571*

content rect
propriété *222*
content size
propriété *207*, *214*
content view
propriété .. *79*, *199*, *202*, *207*, *214*
content view margins
propriété *199*
contents
propriété ... *46*, *65*, *266*, *280*, *307*, *373*, *571*
context
propriété *51*
continuous
propriété *266*, *281*, *516*
control
élément *85*, *210*, *233*
classe *280*
control key down
propriété *51*
control size
énumération *180*
propriété *223*, *266*, *307*
control tint
énumération *180*
propriété *223*, *266*, *307*
control view
propriété *266*
controller visible
propriété *297*
copies on scroll
propriété *202*
corner view
propriété *407*
critical
constante *175*
current cell
propriété *281*, *291*
current column
propriété *291*
current editor
propriété *281*
current item
propriété *275*
current menu item
propriété *302*
current row
propriété *291*
current tab view item
propriété *223*
current text editor
propriété *80*
cursor update type
constante *181*
custom palette mode
constante *180*

D

data
classe *45*
data cell
élément ... *377*, *381*, *385*, *388*
classe *372*
propriété *399*
data column
élément *385*, *389*
classe *376*
data item
élément *373*, *381*, *389*
classe *379*
data representation
event *465*
data row
élément ... *373*, *378*, *381*, *389*
classe *384*

data source
élément *34*, *411*
classe *386*
propriété *275*, *377*, *385*
default entry
élément *73*
classe *45*
default return
constante *174*
default tint
constante *181*
delta x
propriété *51*
delta y
propriété *51*
delta z
propriété *51*
deminiaturized
event *130*
descending
constante *187*
destination window
propriété *477*
dialog ended
event *552*
dialog reply
classe *518*
directory
propriété *530*
display
commande *537*
display alert
commande *540*
display dialog
commande *543*
display panel
commande *547*
displayed cell
propriété *365*
document
élément *34*, *85*
classe *457*
document edited
propriété *80*
document nib name
event *130*
document rect
propriété *203*
document view
propriété *203*, *214*
double clicked
event *349*
double value
propriété *266*, *281*
drag
event *483*
drag entered
event *483*
drag exited
event *484*
drag info
élément *34*
classe *477*
drag updated
event *485*
dragged column
propriété *402*
dragged distance
propriété *402*
drawer
élément *85*
classe *204*
drawer closed
constante *181*
drawer closing
constante *181*
drawer opened
constante *181*

drawer opening
constante *181*
drawer state
énumération *181*
draws background
propriété . *203*, *214*, *223*, *291*, *326*, *331*, *571*
draws cell background
propriété *292*
draws grid
propriété *407*
drop
event *486*
dynamically scrolls
propriété *215*

E

echos bullets
propriété *315*
edge
propriété *208*
editable
propriété . *267*, *286*, *297*, *327*, *399*, *571*
edited column
propriété *407*
edited data column
propriété *407*
edited data item
propriété *394*
edited data row
propriété *408*
edited row
propriété *408*
enabled
propriété .. *267*, *281*, *500*, *521*
enclosing scroll view
propriété *230*
end editing
event *350*
end frame
constante *183*
entry type
propriété *267*
error return
constante *174*
event
élément *34*
classe *50*
event number
propriété *52*
event type
énumération *181*
propriété *52*
excluded from windows menu
propriété *80*
executable path
propriété *39*
expanded
propriété *530*
exposed
event *131*

F

field editor
propriété *571*
file kind
propriété *461*
file name
propriété *460*
file type
propriété *461*
first responder
propriété *80*
first visible column
propriété *365*
flags changed type
constante *181*
flipped
propriété *231*

float value
propriété *267*, *281*
floating
propriété *526*
font
classe *54*
propriété ..*267*, *281*, *521*, *572*
font panel
propriété *31*
font-panel
classe *520*
formatter
classe *56*
propriété *267*, *282*
frameworks path
propriété *39*

G

gave up
propriété *519*
go
commande *334*
go to
énumération *183*
gray bezel frame
constante *184*
gray mode
constante *180*
grid color
propriété *408*
groove border
constante *176*
groove frame
constante *184*

H

has data items
propriété *380*
has horizontal ruler
propriété *215*
has horizontal scroller
propriété *215*, *366*
has parent data item
propriété *380*
has resize indicator
propriété *80*
has shadow
propriété *81*
has sub menu
propriété *500*
has valid object value
propriété *267*
has vertical ruler
propriété *215*
has vertical scroller
propriété *215*, *276*
header cell
propriété *399*
header view
propriété *408*
hidden
propriété *31*
hide
commande *99*
hides when deactivated
propriété *81*
highlight
commande *335*
highlight mode
constante *185*
highlighted
propriété *268*
highlights by
propriété *262*
horizontal line scroll
propriété *215*
horizontal page scroll
propriété *215*
horizontal ruler view
propriété *215*

horizontal scroller
propriété *216*
horizontally resizable
propriété *572*
hsb mode
constante *180*

I

icon image
propriété *31*
id
propriété *61*
identifier
propriété *39*, *399*
idle
event *132*
ignores multiple clicks
propriété *282*
image
élément *34*
classe *58*
propriété . *256*, *268*, *287*, *317*, *478*, *500*
image above
constante *178*
image alignment
énumération *183*
propriété *284*, *287*
image below
constante *178*
image cell
classe *284*
image cell type
constante *179*
image dims when disabled
propriété *262*
image frame style
énumération *184*
propriété *285*, *287*
image left
constante *178*
image location
propriété *478*
image only
constante *178*
image overlaps
constante *178*
image position
propriété *256*, *268*
image right
constante *178*
image scaling
énumération *184*
propriété *285*, *287*
image view
élément *86*, *210*, *233*
classe *285*
imports graphics
propriété *268*, *327*, *572*
increment
commande *336*
increment value
propriété *321*
indentation per level
propriété *394*
indeterminate
propriété *308*
informational
constante *175*
integer value
propriété *268*, *282*
intercell spacing
propriété *276*, *292*, *409*
item
élément *34*
classe *60*
item expandable
event *432*

item for
commande *420*
item height
propriété *276*
item value
event *434*

J

justified text alignment
constante *189*

K

key
propriété *81*
key cell
propriété *292*
key code
propriété *52*
key down type
constante *182*
key equivalent
propriété *256*, *268*, *500*
key equivalent modifier
propriété *256*, *262*, *500*
key up type
constante *182*
key window
propriété *32*
keyboard down
event *133*
keyboard up
event *134*
knob thickness
propriété *317*

L

label
propriété *227*
last column
propriété *366*
last visible column
propriété *366*
launched
event *135*
leading offset
propriété *208*
leaf
propriété *371*
left alignment
constante *183*
left edge
constante *186*
left mouse down type
constante *182*
left mouse dragged type
constante *182*
left mouse up type
constante *182*
left tabs bezel border
constante *189*
left text alignment
constante *189*
level
propriété *81*
line border
constante *176*
line scroll
propriété *216*
list mode
constante *185*
load data representation
event *467*
load image
commande *100*
load movie
commande *104*
load nib
commande *106*
load panel
commande *548*

load sound
commande *107*
loaded
propriété *366*, *371*
localized sort
propriété *388*
localized string
commande *109*
location
propriété *52*, *478*
lock focus
commande *240*
log
commande *111*
loop mode
propriété *297*
looping back and forth playback
constante *186*
looping playback
constante *186*

M

main
propriété *81*
main bundle
propriété *32*
main menu
propriété *32*
main window
propriété *32*
marker follows cell
propriété *394*
matrix
élément *86*, *210*, *234*
classe *289*
matrix mode
énumération *185*
propriété *292*
maximum content size
propriété *208*
maximum size
propriété *82*, *572*
maximum value
propriété *308*, *317*, *321*
maximum visible columns
propriété *366*
maximum width
propriété *399*
menu
élément *303*, *497*
classe *495*
propriété *69*, *268*, *501*
menu item
élément *303*, *497*
classe *499*
miniaturized
event *136*
propriété *82*
minimized image
propriété *82*
minimized title
propriété *82*
minimum column width
propriété *366*
minimum content size
propriété *208*
minimum size
propriété *82*, *572*
minimum value
propriété *308*, *317*, *322*
minimum width
propriété *399*
mixed state
constante *178*
modified
propriété *461*
momentary change button
constante *177*

momentary light button
constante *177*
momentary push in button
constante *177*
mouse down
event *137*
mouse down state
propriété *268*
mouse dragged
event *138*
mouse entered
event *139*
mouse entered type
constante *182*
mouse exited
event *139*
mouse exited type
constante *182*
mouse moved
event *140*
mouse moved type
constante *182*
mouse up
event *141*
moved
event *142*
movie
élément *34*
classe *63*
propriété *298*
movie controller
propriété *298*
movie file
propriété *298*
movie rect
propriété *298*
movie view
élément *86*, *210*, *234*
classe *296*
muted
propriété *298*

N

name
propriété *32*, *61*, *65*, *373*, *377*, *461*
natural text alignment
constante *190*
needs display
propriété *83*, *231*
next state
propriété *269*
next text
propriété *292*, *327*
no border
constante *176*
no frame
constante *184*
no image
constante *178*
no scalling
constante *185*
no tabs bezel border
constante *189*
no tabs line border
constante *189*
no tabs no border
constante *189*
no title
constante *191*
normal playback
constante *186*
null cell type
constante *179*
number of browser rows
event *436*
number of items
event *437*

number of rows
event *439*
number of tick marks
propriété *317*
numerical
constante *188*

O

off state
constante *178*
old style type
constante *176*
on off button
constante *177*
on state
constante *178*
only tick mark values
propriété *317*
opaque
propriété *83*, *231*, *269*
open drawer
commande *240*
open panel
propriété *32*
open untitled
event *143*
open-panel
classe *522*
opened
event *143*
option key down
propriété *52*
other mouse down type
constante *182*
other mouse dragged type
constante *182*
other mouse up type
constante *182*
other return
constante *174*
outline table column
propriété *395*
outline view
élément *210*, *234*
classe *392*

P

page scroll
propriété *216*
pane splitter
propriété *219*
panel
classe *526*
panel ended
event *553*
parent data item
propriété *380*
parent window
propriété *208*
pasteboard
élément *34*
classe *64*
propriété *478*
path
propriété *39*, *366*, *461*
path for
commande *112*
path name
propriété *531*
path names
propriété *523*
path separator
propriété *367*
pause
commande *337*
perform action
commande *337*
periodic type
constante *182*

photo frame
constante *184*
play
commande *338*
playing
propriété *71*, *298*
plays every frame
propriété *299*
plays selection only
propriété *299*
plugin
classe *559*
plugin loaded
event *561*
popup button
élément *86*, *210*, *234*
classe *301*
position
propriété *83*, *231*
poster frame
constante *183*
preferred edge
propriété *209*, *302*
preferred type
propriété *65*
prepare drop
event *488*
pressed
constante *188*
pressure
propriété *52*
previous text
propriété *293*, *327*
primary type
constante *176*
progress indicator
élément *86*, *210*, *234*
classe *306*
prompt
propriété *531*
prototype cell
propriété *293*
pulls down
propriété *302*
push on off button
constante *177*

Q

quicktime movie loop mode
énumération *185*

R

radio button
constante *177*
radio mode
constante *185*
rate
propriété *299*
read from file
event *469*
rectangle edge
énumération *186*
register
commande *116*
regular size
constante *180*
regular square bezel
constante *175*
released when closed
propriété *83*
repeated
propriété *53*
required file type
propriété *531*
resigned active
event *145*
resigned key
event *145*
resigned main
event *146*

resizable
propriété *400*
resized
event *147*
resized column
propriété *402*
resized sub views
event *244*
resource path
propriété *40*
responder
classe *68*
resume
commande *339*
reuses columns
propriété *367*
rgb mode
constante *180*
rich text
propriété *572*
right alignment
constante *184*
right edge
constante *186*
right mouse down
event *147*
right mouse down type
constante *182*
right mouse dragged
event *149*
right mouse dragged type
constante *182*
right mouse up
event *150*
right mouse up type
constante *182*
right tabs bezel border
constante *189*
right text alignment
constante *190*
roll over
propriété *257*, *263*
rounded bezel
constante *175*
row height
propriété *409*
ruler visible
propriété *573*
rulers visible
propriété *216*

S

save panel
propriété *32*
save-panel
classe *529*
scale proportionally
constante *185*
scale to fit
constante *185*
script
propriété *61*
scripts path
propriété *40*
scroll
commande *339*
scroll to location
énumération *186*
scroll view
élément *86*, *210*, *234*
classe *213*
scroll wheel
event *151*
scroll wheel type
constante *182*
scrollable
propriété *269*, *293*
secondary type
constante *176*

secure text field
élément *86*, *211*, *234*
classe *311*
secure text field cell
classe *314*
select
commande *117*
select all
commande *117*
selectable
propriété *269*, *327*, *573*
selected
constante *188*
selected cell
propriété *367*
selected column
propriété *367*, *409*
selected columns
propriété *409*
selected data column
propriété *409*
selected data columns
propriété *409*
selected data item
propriété *394*
selected data items
propriété *395*
selected data row
propriété *410*
selected data rows
propriété *410*
selected row
propriété *410*
selected rows
propriété *410*
selected tab view item
event *245*
selection by rect
propriété *293*
selection changed
event *351*
selection changing
event *352*
send action on arrow key
propriété *367*
sends action when done editing
propriété *269*
separates columns
propriété *367*
separator item
propriété *501*
separator type
constante *177*
sequence number
propriété *478*
services menu
propriété *33*
shadowless square bezel
constante *175*
shared frameworks path
propriété *40*
shared support path
propriété *40*
sheet
propriété *84*
shift key down
propriété *53*
should begin editing
event *353*
should close
event *151*
should collapse item
event *440*
should end editing
event *354*
should expand item
event *441*

should open
event *152*
should open untitled
event *153*
should quit
event *154*
should quit after last window closed
event *155*
should select column
event *442*
should select item
event *443*
should select row
event *444*
should select tab view item
event *246*
should selection change
event *445*
should zoom
event *156*
show
commande *118*
shown
event *157*
shows alpha
propriété *516*
shows state by
propriété *263*
size
propriété *84*, *232*
size to fit
commande *119*
slider
élément *86*, *211*, *234*
classe *315*
small size
constante *180*
smart insert delete enabled
propriété *573*
sort case sensitivity
énumération *187*
propriété *377*
sort column
propriété *388*
sort order
énumération *187*
propriété *377*
sort type
énumération *188*
propriété *377*
sorted
propriété *388*
sound
élément *35*
classe *70*
propriété *257*, *263*
source
propriété *478*
source mask
propriété *479*
spell checking enabled
propriété *573*
split view
élément *86*, *211*, *234*
classe *218*
start
commande *340*
state
propriété ..*209*, *257*, *269*, *501*
step back
commande *341*
step forward
commande *342*
stepper
élément *86*, *211*, *234*
classe *320*
stop
commande *343*

string value
propriété *269*, *282*
sub menu
propriété *501*
super menu
propriété *497*
super view
propriété *232*
switch button
constante *177*
synchronize
commande *343*
system defined type
constante *183*

T

tab key traverses cells
propriété *293*
tab state
énumération *188*
propriété *227*
tab type
propriété *223*
tab view
élément *86*, *211*, *234*
classe *221*
propriété *227*
tab view item
élément *224*
classe *227*
tab view type
énumération *188*
table column
élément *411*
classe *398*
table header cell
classe *401*
table header view
élément *87*, *211*, *235*
classe *402*
table view
élément *87*, *211*, *235*
classe *403*
propriété *400*, *402*
tag
propriété *232*, *269*, *501*
target
propriété *270*, *282*
text
élément *574*
classe *569*
text alignment
énumération *189*
text cell type
constante *179*
text color
propriété *327*, *331*, *573*
text container inset
propriété *573*
text container origin
propriété *573*
text field
élément *87*, *211*, *235*
classe *325*
text field cell
classe *330*
text returned
propriété *519*
text view
élément *87*, *211*, *235*
classe *569*
thick square bezel
constante *175*
thicker square bezel
constante *175*
tick mark above
constante *190*
tick mark below
constante *190*

tick mark left
constante *190*
tick mark position
énumération *190*
propriété *318*
tick mark right
constante *190*
time stamp
propriété *53*
title
propriété .. *84*, *199*, *257*, *270*, *318*, *497*, *501*, *531*
title cell
propriété *199*, *318*
title color
propriété *318*
title font
propriété *199*, *318*
title height
propriété *367*
title position
énumération *190*
propriété *199*
title rect
propriété *200*
titled
propriété *368*
toggle button
constante *177*
tool tip
propriété *232*
top
constante *187*
top alignment
constante *184*
top edge
constante *186*
top left alignment
constante *184*
top right alignment
constante *184*
top tabs bezel border
constante *189*
track mode
constante *185*
trailing offset
propriété *209*
transparent
propriété *257*, *263*
treat packages as directories
propriété *531*
truncated labels
propriété *223*
types
propriété *66*

U

unlock focus
commande *241*
unmodified characters
propriété *53*
update
commande *119*
update menu item
event *506*
update views
propriété *388*
updated
event *158*
user defaults
propriété *33*
user-defaults
classe *72*
uses data source
propriété *276*
uses font panel
propriété *574*
uses ruler
propriété *574*

uses threaded animation
propriété *308*
uses title from previous column
propriété *368*

V

value wraps
propriété *322*
version
propriété *33*
vertical
propriété *219*, *318*
vertical line scroll
propriété *216*
vertical page scroll
propriété *216*
vertical ruler view
propriété *216*
vertical scroller
propriété *216*
vertically resizable
propriété *574*
view
élément *87*, *211*, *235*, *389*
classe *229*
propriété *228*
visible
constante *187*
propriété *84*, *232*
visible document rect
propriété *203*, *216*
visible rect
propriété *232*
volume
propriété *299*

W

warning
constante *175*
was hidden
event *159*
was miniaturized
event *159*
width
propriété *400*
will become active
event *160*
will close
event *161*
will dismiss
event *355*
will display browser cell
event *446*
will display cell
event *447*
will display item cell
event *449*
will display outline cell
event *450*
will finish launching
event *161*
will hide
event *163*
will miniaturize
event *164*
will move
event *164*
will open
event *165*
will pop up
event *356*
will quit
event *166*
will resign active
event *167*
will resize
event *167*
will resize sub views
event *247*

will select tab view item
event *248*
will show
event *169*
will zoom
event *169*
window
élément *35*, *462*
classe *76*
propriété *53*, *233*
windows menu
propriété *33*
wraps
propriété *270*
write to file
event *470*

Z

zoomed
event *171*
propriété *84*